Interreg

EVROPSKÁ UNIE

Rakousko-Česká republika

Evropský fond pro regionální rozvoj

STUDIJNÍ MATERIÁLY PRO OBOR INFORMATIKA

LERNMATERIALIEN FÜR DEN BEREICH INFORMATIK

STUDY MATERIAL FOR THE FIELD OF INFORMATICS

ČESKO / NĚMECKO / ANGLICKÝ
TSCHECHISCH / DEUTSCH / ENGLISCH
CZECH / GERMAN / ENGLISH

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

VŠTE

EVROPSKÁ UNIE

Studijní materiály pro obor informatika

Lernmaterial für den Bereich Informatik

Study material for the field of informatics

Tato publikace vznikla na Vysoké škole technické a ekonomické v Českých Budějovicích a University of Applied Sciences Upper Austria v rámci programu Interreg jako součást projektu „Metodický koncept k efektivní podpoře klíčových odborných kompetencí s využitím cizího jazyka“, registrační číslo 62.

Projekt “CLIL” je financován s podporou Evropské komise, Evropského fondu pro regionální rozvoj (EFRE) a spolkové země Horní Rakousko v rámci programu INTERREG V-A Rakousko-Česká republika 2014-2020.

Diese Publikation entstand in Kooperation zwischen dem Institut für Technik und Wirtschaft in Budweis und der Fachhochschule Oberösterreich im Rahmen des Interreg-Projekts „Methodenkonzept zur effektiven Unterstützung von beruflichen Schlüsselkompetenzen in einer Fremdsprache“, Projektnummer 62.

Das Projekt "CLIL" wurde mit Unterstützung der Europäischen Kommission, des Europäischen Fonds für regionale Entwicklung (EFRE) und des Landes Oberösterreich im Rahmen des Programms INTERREG V-A Österreich-Tschechische Republik 2014-2020 finanziert.

This publication was created in cooperation between the Institute of Business and Technology in České Budějovice and University of Applied Sciences Upper Austria within the framework of the Interreg project “Methodological concept to effectively support key professional competences using foreign language”, reg. No. 62.

The project “CLIL” has been funded with support from the European Commission, the European Fund for Regional development (EFRE), and the Federal State of Upper Austria under the program INTERREG V-A Austria-Czech Republic 2014-2020.

Manažeři projektu / Projektleitung / Project Lead

Mgr. Libuše Turinská
Mag. Dr. Martina Gaisch

Autoři / Autor*innen / Authors

Ing. Bc. Karel Antoš
Ing. Jiří Čejka, Ph.D.
Mgr. Václav Dobiáš
Mgr. Stanislav Jíra
Mgr. Tomáš Náhlík, Ph.D.
MhDr. Helena Pavličíková, CSc.

Editoři / Editor*innen / Editors

Sabrina Menger, MSc.
Victoria Rammer, MMA
Ing. Michal Ruschak

Vydavatel / Herausgeber / Publisher

Vysoká škola technická a ekonomická v Českých Budějovicích
Okružní 517/10
370 01 České Budějovice

Rok vydání / Erscheinungsjahr / Year of publication

2019

ISBN:

INFORMATIKA - ČESKO

Interreg
EVROPSKÁ UNIE
Rakousko-Česká republika
Evropský fond pro regionální rozvoj

ÚVOD

Předkládaná odborná kniha s názvem „Studijní materiály pro obor informatika“ byla připravena v rámci projektu „Metodický koncept k efektivní podpoře klíčových odborných kompetencí s využitím cizího jazyka – CLIL jako výuková strategie na vysoké škole“ realizovaného s finanční podporou Evropské unie, programu INTERREG V-A Rakousko – Česká republika 2014 – 2020.

Projekt je realizován za spolupráce dvou technicky zaměřených vysokoškolských institucí, Vysoké školy technické a ekonomické v Českých Budějovicích, Česká republika, a University of Applied Sciences, Horní Rakousko. Jedním z hlavních výstupů projektu byla příprava odborných didaktických materiálů pro čtyři obory vyučované na partnerských institucích (Informatika, Logistika a doprava, Stavebnictví a Strojírenství), a to ve třech klíčových jazycích: českém, německém a anglickém. Jako výuková metoda byla zvolena metoda CLIL (Content and Language Integrated Learning – obsahově a jazykově integrované učení), kombinující výuku odborného předmětu v kombinaci s výukou cizího jazyka. Připravené materiály tak mají velký význam nejen jako výukový a studijní materiál na odborných vysokých školách, ale poslouží i expertům z konkrétních oborů a zaměstnancům firem působících v přeshraničním regionu, kteří mají možnost zlepšit si své odborné jazykové znalosti.

Na přípravě materiálů se podíleli vyučující z obou partnerských institucí i experti z praxe z obou příhraničních regionů. Materiály z oboru Informatika byly připraveny vyučujícími odborných předmětů. Jejich témata byla vybrána a konzultována ve spolupráci s experty z praxe. Celkově tak bylo vybráno a zpracováno následujících dvanáct témat: Teorie informatiky, Algoritmy a datové struktury, Databáze, Informační a komunikační technologie, Úvod do programování JAVA, Objektově orientované programování JAVA, Úvod do studia médií, IT Security, Sítě, Softwarové inženýrství, Základy webových aplikací, Webové technologie. Rozsah témat byl zvolen tak, aby odpovídal potřebám praxe a zahrnoval co nejširší škálu, od prezentování základů a teorie po konkrétní praktické problémy. Každé z témat je navíc rozděleno do dalších podkapitol. Při výuce i studiu je tak možné prostudovat celý nabízený rozsah i vybrané kapitoly. Materiály jsou dostupné online, každý student i učitel tak má možnost sestavit si obsah kurzu či výuky dle svých konkrétních potřeb.

Jak již bylo zmíněno výše, materiály jsou připravené trojjazyčně. Každé připravené téma bylo následně zpracováno lingvistickými odborníky tak, aby odpovídalo principům metody CLIL a umožnilo tak osvojení si nejen odborných, ale i jazykových znalostí. Znalost cizího jazyka na odborné úrovni se dnes jeví jako klíčová při získání vhodného zaměstnání. Tato publikace tak může posloužit nejen vyučujícím odborných předmětů a studentům odborných vysokých škol, ale i absolventům a zaměstnavatelům a zaměstnancům firem působícím ve výše zmíněných oblastech v přeshraničním regionu i mimo něj, což představuje její značnou přidanou hodnotu.

TEORIE INFORMATIKY

1. Základní matematické pojmy a číselné soustavy

- Data neboli údaje (vjemy), které dokážeme zachytit smysly
- Informace jsou data, kterým rozumíme, mají pro nás smysl, lze je kvantifikovat (měřit) po převedení do číselné podoby
- Informace mohou být: Textové, Obrazové nebo Zvukové.
- Informatika je věda zabývající se uchování, zpracováním využitím údajů a informací. Je částí matematiky a jako prostředek využívá výpočetní techniku.
- Odvětví informatiky:
- Výpočetní technika - zkoumá technické vybavení
- Algoritmizace - navrhování postupů k řešení problémů
- Programování - převádění algoritmů do programovacího jazyka
- Softwarové inženýrství - nauka o vývoji aplikací
- Počítačová grafika - nauka o tvorbě a zpracování 2D i 3D obrazů
- Počítačové modelování a simulace - aplikace matematických modelů na reálné situace
- Formální logika, teorie automatů a formálních jazyků - matematické modely strojů a formálních jazyků pro zápis algoritmů a programů
- Kybernetika a robotika - vývoj strojů schopných samostatné činnosti
- Umělá inteligence - zkoumání procesů lidského myšlení, jejich matematický popis a aplikace při vývoji strojů
- Hardware je technické vybavení počítače, souhrnný název pro veškerá fyzická zařízení, kterými je počítač vybaven
- Software neboli programové vybavení počítač. Lze ho rozdělit na systémový a aplikační.
- Informace uložené v počítači měříme v jednotkách Bit – b – a Byte – B
- Bit je základní a nejmenší jednotka informace má dvě hodnoty: 0, 1
- Byte jako jednotka informace má hodnotu 8bitů.

Jelikož informatika je částí matematiky, používá také některé její pojmy, jako jsou:

- Kartézský součin
- Relace
- Zobrazení
- Dělitelnost
- Největší společný dělitel, nejmenší společný násobek
- Prvočísla

Jelikož počítač ukládá informace v bitech a jeho násobcích, pracuje ve dvojkové číselné soustavě. Je potřeba si říct něco o číselných soustavách.

Číselné soustavy dělíme na poziční a nepoziční.

Poziční číselné soustavy - Hodnota každé číslice dána její pozicí v sekvenci symbolů

Jejich výhodou je velká pružnost a poměrně malá množina číslic, nevýhodou je velmi snadná změna hodnoty čísla pouhým připsáním číslice před původní číslo.

Klíčová charakteristika je Základ, pak váhy jednotlivých číslic jsou mocninami základu.

Příklady:

Jednotková (unární) základ: 1

Dvojková (binární) – základ: 2, dva stavy – ano/ne, 1 bit, symboly: 0, 1

Osmičková (oktalová) – základ: 8, 1 byte (= 8 bitů), symboly: 0, 1, 2, 3, 4, 5, 6, 7

Desítková (dekadická) – základ: 10, přirozená pro lidi (10 prstů), symboly: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (číslice)

Šetnácková (hexadecimální) – základ: 16, 2 byte (= 16 bitů), symboly: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Šedesátková (hexagesimální) – základ: 60, měření času

Nepoziční číselné soustavy - Způsob reprezentace čísel, ve kterém není hodnota číslice dána jejím umístěním v dané sekvenci číslic. Tyto způsoby zápisu čísel se dnes již téměř nepoužívají a jsou považovány za zastaralé, přesto mají jisté výhody jako je jednoduché sčítání a odečítání. Nevýhody jsou, že často neobsahovaly symbol pro nulu a záporná čísla a také dlouhý zápis čísel, která výrazně převyšují hodnotu největšího symbolu soustavy

Příklad: Římské číslice

Převody mezi soustavami

Ze soustavy s vyšším základem

Převádíme-li ze soustavy s vyšším základem do soustavy se základem nižším, pak obvykle používáme algoritmus zapisování zbytků po dělení.

73/2 = 36 + 1 - 1 na 1. místě odzadu
36/2 = 18 + 0 - 0 na 2. místě odzadu
18/2 = 9 + 0 - 0 na 3. místě odzadu
9/2 = 4 +1 - 1 na 4. místě odzadu
4/2 = 2 + 0 - 0 na 5. místě odzadu
2/2 = 1 + 0 - 0 na 6. místě odzadu

1/2 = 0 +1 - 1 na 7. místě odzadu

Ze soustavy s nižším základem

Převod ze soustavy s nižším základem do soustavy s vyšším základem je snazší. Víme totiž, že číslice na n-tém místě (počítáno od 0) reprezentuje číslo vzniklé umocněním základu na *n-tou*. Tyto čísla posčítáme a máme převedeno.

$1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 = 73_{10}$

2. Logika

Logika je věda o správném usuzování, nebo také umění správné argumentace.

Co je to úsudek (argument)?

Na základě pravdivosti premis (předpokladů) P1…Pn lze usoudit, že pravdivý je i závěr Z

Rozeznáváme několik typů argumentační logiky.

Dedukce

Závěr Z logicky vyplývá z předpokladů P1,…,Pn, značíme P1,…,Pn|=Z, jestliže za žádných okolností nemůže nastat případ takový, že předpoklady by byly pravdivé a závěr nepravdivý.

Příklad:

P1: Všichni králíci z klobouku jsou bílí.

P2: Tito králíci jsou z klobouku

Z: ⇒ Tito králíci jsou bílí.

Indukce

Generalizace – od konkrétního k obecnému

Příklad:

P1: Tito králíci jsou z klobouku

P2: Tito králíci jsou bílí.

Z: ⇒ (asi) Všichni králíci z klobouku jsou bílí.

Závěr Z platí jen s určitou pravděpodobností

Abdukce

Vytváříme hypotézy pro pozorované jevy, diagnostika „poruch“

Příklad:

P1: Všichni králíci z klobouku jsou bílí.

P2: Tito králíci jsou bílí.

Z: ⇒ (asi) Tito králíci jsou z klobouku

Závěr Z platí jen s určitou pravděpodobností

Příklad:

P1: Všechny mochomůrky zelené jsou prudce jedovaté

P2: Tato houba je mochomůrka zelená.

Z: Tato houba je prudce jedovatá.

Toto je platná dedukce

Příklad:

P1: Všechny mochomůrky zelené jsou prudce jedovaté

P2: Tato tužka je mochomůrka zelená.

Z: Tato tužka je prudce jedovatá.

Úsudek správný, Závěr nepravdivý ⇒ Alespoň jedna premisa je nepravdivá

Úsudek má správnou logickou formu

Logické spojky – „a“, „nebo“, „jestliže“, „pak“ a jiné mají pevný význam, interpretujeme pomocí nich elementární výroky nebo jejich části (predikáty a funkční termy)

Logika je nástroj, který pomáhá objevovat vztah logického vyplývání, řešit úlohy typu „Co vyplývá z daných předpokladů?“. Pomáhá naší intuici, která může někdy selhat, protože premisy mohou být složitě formulované, zapletené do sebe a do negací, vztah vyplývání není na první pohled patrný.

Příklad:

P1: Všichni muži mají rádi fotbal a pivo

P2: Někteří milovníci piva nemají rádi fotbal

P3: Xaver má rád pouze milovníky fotbalu a pivu

Z: Některé ženy nemá Xaver rád.

Nutně, jsou-li pravdivé všechny předpoklady, pak musí být pravdivý i závěr.

Je úsudek platný?

Jistě, má-li Xaver rád pouze milovníky fotbalu a piva (3.), pak nemá rád některé milovníky piva (ty co nemají rádi fotbal (2.)), tedy nemá rád (dle 1.) některé „ne-muže“, tj. ženy.

Dle definice však platný není

Úsudek je platný, pokud je nutně, tj. za všech okolností (interpretací) kdy jsou pravdivé předpoklady, pravdivý i závěr.

Ale: v našem příkladu ta individua, která nejsou muži, by nemusela být interpretována jako ženy. Chybí zde premisa, že „kdo není muž, je žena“, podobně ještě potřebujeme premisu „kdo je milovník něčeho, ten to má rád“.

Musíme uvádět všechny předpoklady nutné pro odvození závěru

P1: Všichni muži mají rádi fotbal a pivo.

P2: Někteří milovníci piva nemají rádi fotbal.

P3: Xaver má rád pouze milovníky fotbalu a piva.

P4: Kdo není muž, je žena.

P5: Kdo je milovník něčeho, ten to má rád.

Z: Některé ženy nemá Xaver rád.

Nyní je úsudek správný, má platnou logickou formu.

Závěr logicky vyplývá z předpokladů (je v nich „informačně, dedukčně obsažen“).

Vlastnosti logiky

Platný (správný) úsudek může mít nepravdivý závěr

Využití: Důkaz AD ABSURDUM

Monotónnost

Je-li úsudek platný, rozšíření množiny předpokladů o další předpoklad nevede ke změně platnosti úsudku

Ze sporných předpokladů vyplývá jakýkoli závěr

Reflexivita, Tranzitivita

Platná úsudková schémata

1. $A \supset B, A| = B$

modus ponens

P_1: Žádné prvočíslo není dělitelné 3

P_2: 9 je dělitelné 3

Z: 9 není prvočíslo

2. $A \supset \neg B, B| = \neg A$

modus ponens + transpozice

P_1: Všichni lidé jsou rozumní

P_2: Kámen není rozumný

Z: Kámen není člověk

3. $A \supset B, \neg B| = \neg A$

modus ponens + transpozice

P_1: Je-li 12 prvočíslo nebo není dělitelné 3

P_2: 12 je dělitelné 3

Z: 12 není prvočíslo

4. $\neg A \vee \neg B, B| = \neg A$

eliminace disjunkce

P_1: 12 není prvočíslo nebo není dělitelné 3

P_2: 12 je dělitelné 3

Z: 12 není prvočíslo

3. Množiny a relace

3.1. Množiny

Definice – souhrn objektů, které jsou přesně určené a rozlišitelné a tvoří součást světa našich představ a myšlenek; tyto objekty nazýváme prvky množiny.

Pro množiny používáme následující symboliku:

- Množiny: A, B, C…
- Prvky: a, b, c…
- Prvek množiny: $a \in A$
- Pro všechna x platí P: $\forall x : P$
  - Existuje x, pro které platí P: $\exists x : P$
  - Existuje právě jedno x tak, že platí P: $\exists! x : P$
- Konjunkce, disjunkce, negace: $\wedge$, $\vee$, $\neg$
- Implikace, ekvivalence: $\Rightarrow$, $\Leftrightarrow$
- Sumace, multiplikace: $\sum$, $\prod$

Množina může být určena

- Výčtem prvků: $A = \{1,2,3,4,5\}$
- Pomocí charakteristické vlastnosti: $A = \{a \in N | a < 6\}$

Definujeme prázdnou množinu: $\{\emptyset\}$, která neobsahuje žádné prvky.

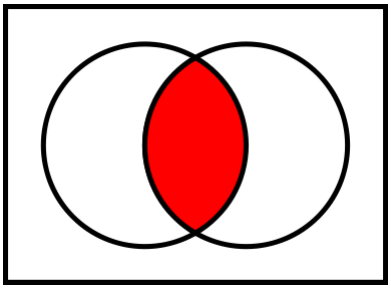
Počet prvků množiny určuje – **mohutnost** či **kardinalita** - $|A|$, kde A je množina.

Rozeznáváme množiny:

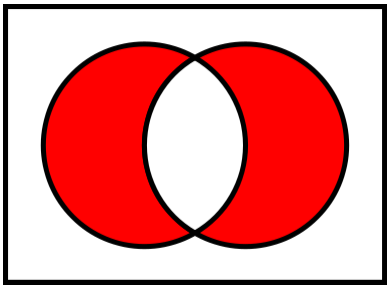
- Konečné
- Nekonečné
- Spočetné
- Kontinuum

Základní operace s množinami:

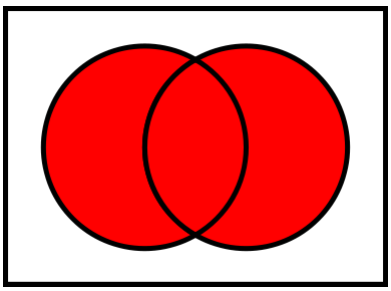
Průnik: $A \cap B$



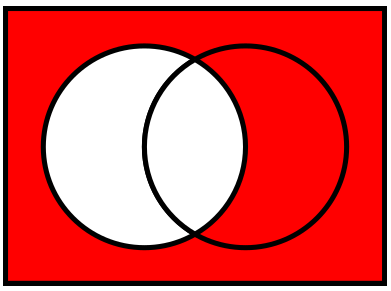
Symetrická diference: $A \triangle B$



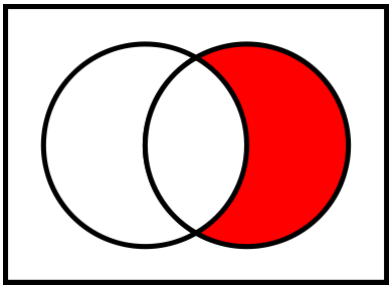
Sjednocení: $A \cup B$



Doplněk A v U: $A^c = U \setminus A$



Rozdíl množin: $B \setminus A$



Podmnožina (inkluze – opak exkluze): $A \subseteq B$

$(\forall x)(x \in A \Rightarrow x \in B)$

Potenční množina – množina všech podmnožin

$$P(\emptyset) = \emptyset$$

$$P(\{a\}) = \{\emptyset, \{a\}\}$$

$$P(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$$

3.2. Relace

Definice: n-ární relací mezi množinami A_1, A_2, A_3, …, A_n, kde $n \in N$, rozumíme libovolnou

podmnožinu kartézského součinu n množin.

Kartézský součin (diskrétní součin)

Množina $X \times Y$, která obsahuje všechny uspořádané dvojice, kde první položka je z X a druhá položka je z Y

A\B	1	2	3
X	(X,1)	(X,2)	(X,3)
Y	(Y,1)	(Y,2)	(Y,3)
Z	(Z,1)	(Z,2)	(Z,3)

Relace lze klasifikovat podle arity na:

Unární – každá podmnožina množiny N

Příklad: je kladné číslo, je pravdivý/nepravdivý výrok

- Binární – každá množina uspořádaných dvojic $[x; y] \in M^2$
- Př.: je větší než, je menší než, je podmnožinou
- Ternární – každá množina uspořádaných trojic $[x; y; z] \in M^3$.
- Př.: leží mezi
- N-ární – každá množina uspořádaných n-tic z M^n.

Operace s relacemi

Kromě klasických množinových operací (všechny relace musí mít stejnou aritu), zavádíme inverzní relaci jako: R^{-1}: $\forall$ a $\in M_1$ $\forall$ b $\in M_2$:[a,b] $\in$ R $\Leftrightarrow$ [b,a] $\in R^{-1}$ a složenou relaci.

Binární relace na množině je:

- **reflexivní**, pokud pro všechna a ∈ M platí, že [a, a] ∈ R.
- **symetrická**, pokud pro všechna a, b ∈ M platí, že je-li [a, b] ∈ R, pak i [b, a] ∈ R.
- **antisymetrická**, pokud pro všechna a, b ∈ M platí, že pokud je [a, b] ∈ R a zároveň je i [b, a] ∈ R, pak a = b.
- **transitivní**, pokud pro všechna a, b , c ∈ M platí, že pokud [a, b] ∈ R a zároveň [b, c] ∈ R, pak i [a, c] je v R.

Dvě speciální relace

- **Ekvivalence** je relace, která je reflexivní, symetrická a transitivní.
- **Uspořádání** je relace, která je reflexivní, antisymetrická a transitivní.

4. Relační struktury

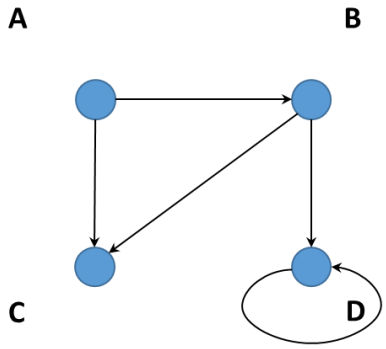
Relační strukturou myslíme matematickou strukturu, která kromě nosné množiny obsahuje jednu či více relací (které mohou mít libovolnou, i navzájem různou aritu). Patří sem orientované grafy a uspořádané množiny (včetně jejich speciálních případů, množin uspořádaných lineárně nebo dobře).

Orientované grafy

Orientovaný graf *G* je dvojice (*V*, *E*), *kde E* je podmnožina kartézského součinu V × V, $E \subseteq V \times V$

Prvky *E* nazýváme **šipky** nebo **orientované hrany**. Orientovaná hrana *e* má tvar (*x*, *y*). Říkáme, že tato orientovaná hrana vychází z *x* a končí v *y*.

Orientované grafy jsou reprezentované pomocí matice sousednosti, nebo seznamem vrcholů a hran.



Matice sousednosti:

		KAM			
		A	B	C	D
ODKUD	A	0	1	1	0
	B	0	0	1	1
	C	0	0	0	0
	D	0	0	0	1

Pro orientované grafy nemusí být symetrická, 1 kudy vede hrana, 0 kde hrana nevede

Množina vrcholů a hran: G: (A, [A,B]), (A, [A,C]) (B, [B,C]), (B, [B,D]), (D, [D,D])

Uspořádané množiny

Lineární uspořádání (= úplné uspořádání) je definované tak, že:

Každé dva prvky lineárně uspořádané množiny jsou porovnatelné, tj. existuje relace *R* na množině ***X*** a *a, b, c* ∈ ***X*** taková, že platí:

Tranzitivita: $(\forall x, y, z \in A)\big((xRy \wedge yRz) \Rightarrow xRz\big)$

Slabá antisymetrie: $(\forall x, y \in A)\big((xRy \wedge yRx) \Rightarrow x = y\big)$

Trichotomie: $aRb \vee bRa \vee a = b$

Příklad: uspořádání na množině přirozených i reálných čísel

Dobré uspořádání je definované tak, že:

Množina ***S*** je dobře uspořádaná právě tehdy, když má každá neprázdná část uspořádané množiny S nejmenší prvek

Příklad: Přirozená čísla

Uspořádaná množina je taková, na níž je definováno uspořádání.

Uspořádání je binární relace *R*, která je reflexivní, tranzitivní a slabě antisymetrická

$(\forall x \in A)(xRx)$ reflexivita – každý prvek je v relaci *R* sám sebou

$(\forall x, y, z \in A)\big((xRy \wedge yRz) \Rightarrow xRz\big)$ tranzitivita - pokud je prvek množiny v uspořádání mezi jinými dvěma prvky, jsou tyto dva rovněž srovnatelné

$(\forall x, y \in A)\big((xRy \wedge yRx) \Rightarrow x = y\big)$ slabá antisymetrie (neexistují cykly v uspořádání)

= neostré uspořádání

Ostré uspořádání – reflexivita nahrazena antireflexivitou – žádný prvek není sám sebou $(\forall x \in A)\big(\neg(xRx)\big)$

5. Zobrazení

Zobrazení je speciální příklad binární relace, u které je zajištěna jednoznačnost obrazu ke každému vzoru.

V případě, že se jedná o binární relaci na číselné množině, zobrazení pak označujeme jako funkci.

Definice:

Zobrazení f z množiny A do množiny B, je taková binární relace, pro kterou platí, že každému prvku x z A přiřazuje nejvýše jeden prvek y z B tak, že $[x, y] \in f$

Značení: $y = f(x)$ $\quad f: x \to y$

Důležité pojmy: $y = f(x) \in B,\ x \in A$

y – obraz prvku x v zobrazení f, hodnota zobrazení f v bodě x

x – vzor prvku y v zobrazení f

Množina prvků $x \in A$, pro které existuje právě jeden takový prvek $y \in B$, že $y = f(x)$, se nazývá definičním oborem zobrazení f

Množina prvků $y \in B$, pro které existuje alespoň jeden takový prvek $x \in A$, že $f(x) = y$, se nazývá oborem hodnot zobrazení f

Rozlišujeme několik základních typů zobrazení:

Zobrazení v množině (zobrazení na množině):

$$f: A \to B,\ A = B$$

Množiny do množiny – celá množina A definičním oborem

$$f: A \to B,\ \ D(f) = A$$

Z množiny na množinu (surjekce) – celá množina B je oborem hodnot

$f: A \to B,\ \ H(f) = B, \forall b \in B: \exists a \in A: f(a) = b$

Množiny na množinu

$f: A \to B,\ \ D(f) = A \wedge H(f) = B,$

$\forall a \in A: \exists! b \in B: b = f(a) \wedge \forall b' \in B: \exists! a' \in A: f(a') = b'$

Prosté (injekce)

$$\forall b \in B: \exists! a \in A: f(a) = b$$

Vzájemně jednoznačné (bijekce) - Prosté zobrazení množiny A na množinu B (je injektivní

a surjektivní zároveň)

$$f: A \leftrightarrow B$$

Inverzní – Existuje pouze pro prostá zobrazení

Je-li $f: A \to B$ prosté, pak zobrazení

$f^{-1}: B \to A$, které $\forall b \in H(f)$: $f^{-1}(b) = a \in D(f): y = f(x)$, se nazývá inverzní

$$D(f^{-1}) = H(f),\ f^{-1}(y) = x \Leftrightarrow f(x) = y$$

Zobrazení lze také skládat:

Množiny ***A***, ***B***, ***C***

Zobrazení: $f: A \to B$ a $g: B \to C$

Složené zobrazení: $h: A \to C,\ h = g \circ f,\ h(a) = g\big(f(a)\big)\ \forall a \in A$

Není obecně komutativní, ale je asociativní

Inverze složeného zobrazení: $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$

6. Boolova algebra

Booleva algebra je definována jako: Šestice $(A, \wedge, \vee, -, 0, 1)$, kde A je neprázdná množina, $0 \in A$ – nejmenší prvek, $1 \in A$ – největší prvek, $-$ unární operace (doplněk, komplement), $\wedge$ - průnik, logický součin, $\vee$ - spojení, logický součet.

Je postavena na axiomech:

Komutativita

$$x \vee y = y \vee x$$
$$x \wedge y = y \wedge x$$

Distributivita

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Neutralita 0 a 1

$$x \vee 0 = x$$
$$x \wedge 1 = x$$

Komplementarita

$$x \vee -x = 1$$
$$x \wedge -x = 0$$

A má následující vlastnosti:

- absorpce: $x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$
- agresivita nuly: $x \wedge 0 = 0$
- agresivita jedničky: $x \vee 1 = 1$
- idempotence: $x \vee x = x$, $x \wedge x = x$
- absorpce negace: $x \vee (-x \wedge y) = x \vee y$, $x \wedge (-x \vee y) = x \wedge y$
- dvojitá negace: $-(-x) = x$
- De Morganovy zákony: $-x \wedge -y = -(x \vee y)$, $-x \vee -y = -(x \wedge y)$
- *0* a *1* jsou vzájemně komplementární: $-0 = 1$, $-1 = 0$

Základní jednovstupové operace:

- ID (identita)
- NOT (negace)
- Dvouvstupové základní operace.
- OR (logický součet)
- AND (logický součin)

Pravdivostní tabulka pro základní operace:

A	B	ID(A)	ID(B)	NOT(A)	NOT(B)	A OR B	A AND B
0	0	0	0	1	1	0	0
0	1	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	1	1	1	0	0	1	1

Odvozené dvouvstupové operace:

- NOR (Not OR – negace součtu)
- NAND (Not AND – negace součinu)
- Implikace ⇒ (při splněném předpokladu A vrací B, nebo z nesplněného předpokladu vyplývá cokoli a vrací 1)
- Ekvivalence EQ ⇔ (shodnost vstupů)
- XOR (Exkluzivní OR – unikátnost vstupů)

Pravdivostní tabulka pro odvozené dvouvstupové operace:

A	B	NOR	NAND	⇒	⇔	XOR
0	**0**	1	1	1	1	0
0	**1**	0	1	1	0	1
1	**0**	0	1	0	0	1
1	**1**	0	0	1	1	0

Mějme výraz: A (B+NOT(C))

Tento výraz lze zapsat pomocí:

- Pravdivostní tabulky:

i	A	B	C	A(B+NOT(C))
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

- Algebraického výrazu:

A (NOT (B) (NOT (C)) + AB (NOT (C)) + ABC

Což je logický součet všech výrazů dávajících výsledek 1

- Množinou indexu stavů a to tak, že řádky v pravdivostní tabulce očíslujeme

od 0 (viz. Sloupec *i*)

A (B+NOT(C)) = {4, 6, 7}

Pro minimalizaci algebraického výrazu lze využít Karnaughovu mapu.

Vyplňuje se tak, že sloupce (řádky), nad kterými je uvedená proměnná, jsou pro ni 1

V tabulce jsou hodnoty uvedené v pořadí ABC

			B	
		C		
	000	001	011	010
A	100	101	111	110

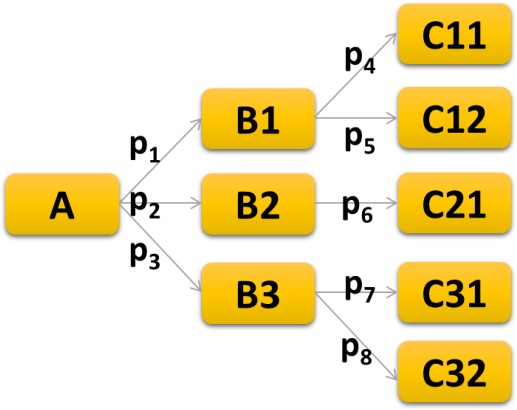
Př: Výraz: A(NOT(B)(NOT(C)) + A(NOT(B))C + ABC = A(NOT(C)) + AB = A(B + NOT(C))

			B	
		C		
	0	0	0	0
A	1	0	1	1

7. Pravděpodobnost a statistika

Rozdíl mezi determinismem a stochasticitou je v tom, že u determinismu je předem znám výsledek. Z A do B jdu se 100% pravděpodobností (vždy jdu touto cestou, jiná možnost není).

Naproti tomu, u stochastický jevů se vše děje jen s určitou pravděpodobností, z A do B1 jdu s pravděpodobností p_1. Ale také můžu jít do B2 s pravděpodobností p_2 a do B3 s p_3. To znamená, že znám jen pravděpodobnost s jakou dojde k danému výsledku. V daném jevu hrají roli náhodné procesy. Pokud se podíváme na obrázek, součet p1, p2 a p3 se musí rovnat 1, neboli z A do nějakého B přejdu se 100% pravděpodobností. Stejně tak pravděpodobnost p6 (B2-> C21) je rovna jedné.



Pravděpodobnost je definována takto:

Mějme experiment E, množinu podmínek experimentu Π a množinu všech možných výsledků F. F obsahuje události 0, A, B, …, Ω, kde 0 je událost, která nikdy nenastane a Ω událost, která nastane vždy. Experiment lze tedy zapsat jako E=(Ω,F). Množina výsledků experimentů opakovaných za stejných podmínek je N, jednotlivé výsledky jsou 1…n a I je podmnožina výsledků. Podmnožina *A(I)* je podmnožina *I* taková, že výsledkem experimentu je událost *A*. Počet výsledků v podmnožině *A(I)* je $n_A(I)$ a množina všech výsledků je $\Omega(I)$. Pak pro všechny podmnožiny *I* platí $n_A(I) \approx P(A) \cdot n_\Omega(I)$, kde *P(A)* je pravděpodobnost jevu *A* Neboli:

$$P(A) \approx \frac{n_A(I)}{n_\Omega(I)}$$

Lze definovat i relativní četnost jevu *A* jako: $\frac{n_A(I)}{n_\Omega(I)}$

Vlastnosti pravděpodobnosti – pravděpodobnost jevu A je vždy mezi 0 a 1 včetně obou krajních hodnot.

Pravděpodobnost, že jev A nenastane, neboli negace jevu A je 1-P(A).

Jestliže jevy A a B nenastávají současně, pak P$(A \cap B) = \emptyset$

Pravděpodobnost, že nastane jev *A*, nebo jev *B*: $P(A \cup B)$ je $P(A \cup B) = P(A) + P(B)$

Podmíněná pravděpodobnost – pravděpodobnost jevu A, když nastane jev B: $P(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}$

$$(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Bayesův teorém: $P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$

Statistika je soubor konceptů, pravidel a postupů, které nám pomáhají organizovat numerické informace, porozumět technikám v pozadí a činit informovaná rozhodnutí.

Statistické poznatky jsou aplikovány na data, což jsou fakta a informace pocházející z pozorování (ideálně z experimentů)

Data dělíme na číselná (kvantitativní) data, která jsou výsledkem měření, a kategorická (kvalitativní) data, rozdělená do skupin na základě společných vlastností.

Základními statistickými veličinami jsou střední hodnota (vážený průměr), medián, což míra centrální tendence, prostřední hodnota v daném souboru a modus jako nejčastější hodnota. Dále vypočítáváme rozptyl, neboli varianci, a z něj směrodatnou odchylku jako odmocninu. Dále můžeme zjišťovat kovarianci, která určuje, jak moc se spolu dvě hodnoty mění. A pak šikmost a špičatost, které vypovídají o rozložení hodnot daného souboru.

Rozložení hodnot souboru nám popisuje takzvané rozdělení. Jedním ze základních rozdělení je Gaussovo (normální) rozdělení. To je symetrické, má stejnou střední hodnotu, modus a medián, šikmost a špičatost jsou nulové.

Jednou za základních vět statistiky je Centrální limitní věta, která Vyjadřuje fakt, že souhrn mnoha nezávislých náhodných veličin bude mít tendenci se rozptylovat v malém souboru distribučních funkcí.

8. Teorie informace

Teorie informace se zabývá měřením, přenosem, kódováním, ukládáním a následným zpracováním informací z kvantitativního hlediska. Jejím zakladatelem je C.E. Shannon.

Informace

To, co si vyměňujeme s vnějším světem, když se mu přizpůsobujeme a působíme na něj svým přizpůsobováním

Obsah zprávy, sdělení, objasnění, vysvětlení, poučení.

Údaje, čísla, znaky, povely, instrukce, příkazy, zprávy apod. Za informace považujeme také podněty a vjemy přijímané a vysílané živými organismy.

Velikost informace je dána uspořádaností (neurčitostí) soustavy prvků (systému).

Informace o systému je tím větší, čím je pravděpodobnost výskytu jednotlivých jeho stavů menší. Informace je větší, obsahuje-li zpráva něco nového, co předtím nebylo známo, nebo co nelze snadno uhodnout (je málo pravděpodobné)

Pro předání informace je potřeba

Nějaký způsob kódování = převod do vhodných signálů nebo symbolů

Signál – fyzikální veličina nesoucí informaci

Zpráva – vyjádření informace pomocí posloupnosti symbolů (znaků)

Zpráva jako taková má tři části:

- Syntaxe – skladba
- Syntaktický obsah
- Může existovat i sám bez sémantického a pragmatického obsahu
- Týká se vzájemného uspořádání znaků jako nositelů informace
- Popisuje kvantitativní stránku informace
- Sémantika – informační obsah
- Sémantický obsah

Významová stránka zprávy, nedá se měřit

Zpráva se stejnou velikostí informace může být zapsána v různých jazycích

Popisuje kvalitativní stránku informace

Důležitost – pragmatický obsah

Určuje významnost (důležitost, užitečnost, cennost) sdělení a prioritu

Kvalitativní stránka informace

Informační obsah zprávy se měří pomocí entropie.

Informace je míra množství neurčitosti nebo nejistoty o nějakém náhodném ději odstraněná realizací tohoto děje.

Míru neurčitosti určíme jako entropii systému

$$H(X) = -\sum_{i=1}^{n} p(x_i)p(x_i)$$

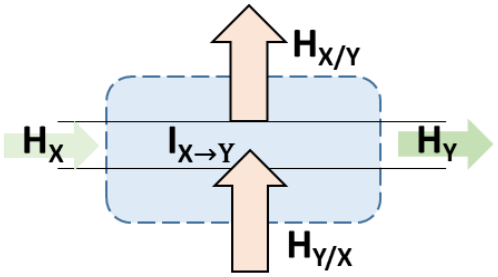
Informace o systému X lze získat:

- Přímým pozorováním
- Zprostředkovaně pomocí systému Y (systém X je pro nás nedostupný)

Stav systému Y nemusí být nutně totožný se stavem systému X (text telegramu v jednom městě X, ve druhém městě Y)

Rozdíly dvojího druhu

- Některé stavy systému X se zobrazí do jednoho stavu Y (Y nedokáže rozlišit jemnosti v X, Y je hrubší než X)
- Chybami při přenosu mezi X a Y – např.: šum
- Pomocí kanálu



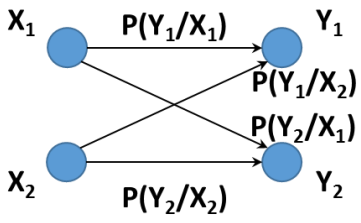
$H_{X/Y}$ – průměrná ztracená informace

$H_{Y/X}$ – průměrná rušivá informace

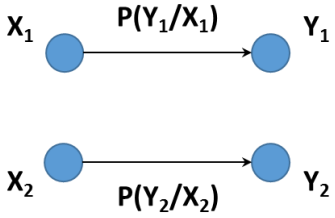
$I_{X\rightarrow Y}$ – vzájemná informace

Pro přenos informace pomocí kanálu, rozlišujeme dva typy kanálů:

Hlukový (šumový):



Bezhlukový:



$P(Y_1/X_1)$ pravděpodobnost, že při vyslání prvku X_1 obdržíme Y_1

$P(Y_2/X_2)$ pravděpodobnost, že při vyslání prvku X_2 obdržíme Y_2

$P(Y_1/X_2)$ pravděpodobnost, že při vyslání prvku X_2 obdržíme Y_1

$P(Y_2/X_1)$ pravděpodobnost, že při vyslání prvku X_1 obdržíme Y_2

Propustnost (kapacita) kanálu – schopnost kanálu přenášet informace. Maximální informace, která může být přenesená za jednotku času.

$C = B\left(1 + \frac{S}{N}\right)$, kde B je šířka kanálu, S je signál, N je šum.

Vzorkovací teorém

Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byla vzorkovací frekvence vyšší než dvojnásobek nejvyšší harmonické složky vzorkovaného signálu.

Minimální možná délka signálových prvků

$\tau_0 = \frac{1}{2F_m}$, F_m je mezní frekvence, šířka pásma

9. Aplikace teorie informace

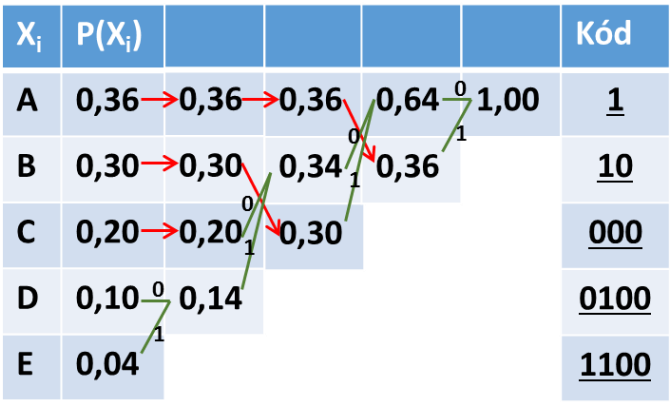
Aplikacemi teorie informace jsou například – Huffmanovo kódování, aritmetické kódování, LZW, Zpracování formátů (komprese) – JPEG, MP3, TIFF, Kódy pro detekci a opravu chyb, Statistické aplikace či Princip minimální deskripční délky (MDL).

Huffmanovo kódování je algoritmus pro bezztrátovou kompresi dat. Konvertuje znaky vstupního souboru do bitových řetězců různé délky. Nejfrekventovanější znaky do bitových řetězců s nejkratší délkou (i 1bit). Nejméně frekventované do delších řetězců (mohou být i delší než 8 bitů).

Má dvě fáze

- Projde soubor a vytvoří statistiku
- Vytvoří binární strom ke kompresi dat
- Kód je tvořen od listů ke kořenu.

Příklad:



Shannonovo-Fanovo kódování je statistická metoda bezeztrátové komprese. Od Huffmanova kódování se liší pouze konstrukcí binárního stromu. Množina znaků je rekursivně dělena vždy na dvě přibližně stejně velké podmnožiny. Jedné podmnožině je pak v kódu přiřazena binární 1 a druhé 0. Kód je tedy konstruován od kořene k listům, na rozdíl od Huffmanova kódování, jehož kód je tvořen od listů ke kořenu, nemusí být optimální.

Příklad:

znak	p(x)	s	skupiny				Kód
A	0,36	1	0				0
B	0,3	0,64	1	0			10
C	0,2	0,34		1	0		110
D	0,1	0,14			1	0	1110
E	0,04	0,04				1	1111

Aritmetické kódování je bezztrátová komprese dat s proměnlivou délkou kódového slova. Zakóduje celý text do jednoho čísla, zlomku $n \in <0;1)$. Je velice náročné na

počítání s reálnými čísly. Při delší zprávě bychom totiž již nebyli schopni dosahovat potřebných přesností, a některé subintervaly by nám mohly začít splývat. Používá komprimace po blocích, kdy bloky jsou tak velké, aby zajišťovaly při rozdělování dostatečnou přesnost.

Informaci lze při přenosu zabezpečit a to pro detekci a opravu chyb a proti neoprávněnému čtení.

Kódy pro detekci a opravu chyb

Paritou - Ke každému úseku dat je připojen další bit, který svou hodnotou doplňuje počet binárních jedniček na počet lichý nebo sudý (sudá/lichá parita)

Sudá: 10011011 10010101 →100110111 100101010

Lichá: 10011011 10010101 →100110110 100101011

CRC – kontrolní součet – Data se rozdělí na úseky požadované délky (8, 16, 32 bitů) a tyto úseky se sečtou po bitech bez přenosu. Vzniklý úsek dat se připojí k datům přenášeným.

00001101	Data
00010000	
01000100	
10010000	
11110001	**CRC**

Hammingův kód je lineární kód používaný v oblasti telekomunikací pro detekci až dvou chybných bitů nebo pro opravu jednoho chybného bitu

Algoritmus:

Všechny bitové pozice, jejichž číslo je rovné mocnině 2, jsou použity pro paritní bit (1, 2, 4, 8, 16, 32, ...).

Všechny ostatní bitové pozice náleží kódovanému informačnímu slovu (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, ...).

Každý paritní bit je vypočítán z některých bitů informačního slova. Pozice paritního bitu udává sekvenci bitů, které jsou v kódovém slově zjišťovány a které přeskočeny.

Pro paritní bit p_1 (pozice 1) se ve zbylém kódovém slově 1 bit přeskočí, 1 zkontroluje, 1 bit přeskočí, 1 zkontroluje, atd.

Pro paritní bit p_2 (pozice 2) se přeskočí první bit, 2 zkontrolují, 2 přeskočí, 2 zkontrolují, atd.

Pro p_3 (pozice 4) se přeskočí první 3 bity, 4 zkontrolují, 4 přeskočí, 4 zkontrolují, atd.

Proti neoprávněnému čtení

- Šifrování – Kryptografie, Historické
- Stenografie – ukrývání zprávy
- Substituční šifra – nahrazení každého znaku jiným podle nějakého pravidla
- Posun písmen – Caesarova šifra – každé písmeno abecedy posunuto o pevný počet pozic
- Tabulky záměn – záměně znaku za jiný bez jakékoli vnitřní souvislosti či na základě znalosti klíče
- Vigenerova šifra – používá heslo, jehož znaky určují posunutí otevřeného textu a to tak, že otevřený text se rozdělí na bloky znaků dlouhé stejně jako heslo a každý znak se sečte s odpovídajícím znakem hesla
- Vermanova šifra – jedinou známou šifru, o níž bylo dosud exaktně dokázáno, že je nerozluštitelná, je založená na sčítání písmen otevřeného textu a hesla, avšak heslo je blok náhodně zvolených dat o stejné velikosti, jako je otevřený text

Transpoziční mřížka

- Skytalé – druh šifrování, který se skládá z válce a na něm navinutém papyru či pergamenu na kterém je napsaný vzkaz - Moderní
- Symetrické šifry – konvenční šifra, používá k šifrování i dešifrování jediný klíč, DES (Data Encryption Standard), současný standard AES (Advanced Encryption Standard)
- Asymetrické šifry – pro šifrování a dešifrování používají odlišné klíče (soukromý a veřejný klíč), RSA, PGP (OpenPGP), používá se pro elektronický podpis
- Hašovací funkce – jednosměrná matematická funkce, která se používá např. pro zajištění integrity dat.

Podepisování

- Elektronický podpis – Označení specifických dat, které v počítači nahrazují klasický vlastnoruční podpis, respektive ověřený podpis. Je připojen k datové zprávě nebo je s ní logicky spojen, umožňuje ověření totožnosti podepsané osoby ve vztahu k datové zprávě. Elektronický podpis je prostředek k tomu, jak ověřit totožnost odesílatele.

10. Teorie složitosti

Teorie složitosti se zaměřuje na klasifikaci výpočetních problémů dle jejich vlastní složitosti a určení vztahů mezi třídami. Problém je úkol, který lze vyřešit na počítači. Problém, se považuje za těžký, pokud jeho řešení potřebuje značné zdroje bez ohledu na to, jaký algoritmus je použit. Formalizuje tento přístup, zavádí výpočetní modely, na nichž studuje a kvantifikuje množství potřebných zdrojů pro řešení problémů (čas, paměť). Další míry složitosti, jako množství komunikace, počet hradel obvodu, počet přístupů do cache a počet procesorů. Jeden z cílů – určit praktické limity toho, co počítače dokážou spočítat a co ne.

Analýza algoritmů vs. teorie složitosti vs. teorie vyčíslitelnosti

- Analýza algoritmů se zabývá množstvím potřebných zdrojů, které potřebuje konkrétní algoritmus
- Teorie složitosti zjišťuje obecnější otázky týkající se všech algoritmů, které se dají použít pro řešení určitého problému. Snaží se klasifikovat problémy podle daných omezení na dostupné zdroje, zavádí omezení na dostupné zdroje
- Teorie vyčíslitelnosti se ptá, jaké problémy lze, v principu, vyřešit algoritmicky.

Základem je výpočetní problém.

Zadání problému: Nekonečná sbírka příkladů a jejich řešení pro danou situaci, zadání vstupu, tzv. instanci problému nelze zaměnit s problémem samotným. Problém je třeba řešit bez ohledu na jeho zadání

Př.: Test na prvočísla – Zadání – číslo; Výstup – Ano/Ne

Problém lze zadat několika způsoby: Nejzákladnějším způsob je v podobě vyslovené věty. Pro řešení počítačem je nutný překlad do jazyka počítačů. Matematické úlohy v oblasti grafů se zadávají na příklad pomocí matic sousednosti.

Rozhodovací problém je základní typ problémů teorie složitosti, má pouze dva výstupy Ano/Ne. V jazyku, kde členy jazyka jsou případy s odpovědí ANO a ostatní členy s odpovědí NE. Je třeba pomocí algoritmu rozhodnout, jakým způsobem zadaný vstup s tímto jazykem odpovídá. Jestliže algoritmus vrátí ANO pak je vstup přijat, jinak odmítnut. Algoritmus počítá charakteristickou funkci jazyka.

Formální jazyky (a problémy nad nimi) jsou nad nějakou abecedou, která nemusí být nutně binární.

Problém funkce je výpočetní problém, kde jeden výstup totální funkce platí pro všechny možné vstupy, ale je složitější než výstup rozhodovací funkce. Problém funkce je bohatší na výsledky než rozhodovací problém. Lze ho přepracovat také na rozhodovací problém a to takto: chceme vynásobit dvě čísla, vstupem rozhodovacího problému je tedy trojice

(a, b, c) a odpověď ANO se vrací pouze pro trojice, kde platí *a*b=c*

Pro teorii složitosti je potřeba měřit velikost vstupu. Velikost vstupu závisí na množství času pro zpracování algoritmem. Tyto dílčí problémy, kterým může být i prostor potřebný pro řešení zpracovává samostatný algoritmus a vše souvisí také s velikostí vstupu v bitech. Teorii složitosti se zajímá o způsob, jakým se algoritmus vypořádá s velikostí vstupu.

Např.: řešení souvislého grafu o n hranách v porovnání s grafem o $2n$ hranách. Jestliže je vstupem n pak čas potřebný pro výpočet je funkce $\tau(n)$. Pokud $\tau(n)$ je polynomiální hovoříme o polynomiálním algoritmu.

Cobhamova teze – problém lze vyřešit v polynomiálním čase, pokud pro něj existuje algoritmus, který zpracuje n bitový vstup v čase $\tau(n^c)$, kde c je konstanta závisející na problému, nikoliv jeho vstupu.

Měříme se však nejen vstupy, ale i zdroje, ať už pro konkrétní algoritmus nebo pro nějaký problém.

Obecně pro data velikosti n a nikoli pro konkrétní hodnotu vstupu k (velikosti n), obvykle pro všechny možné (počtem nekonečné) velikosti → složitost odhadujeme a to obvykle asymptoticky. V závislosti na zdrojích se měří:

- Spotřebovaný čas (v krocích)
- Paměť (v bitech/bajtech/buňkách)
- Pakety (rámcově v rámcích)
- Cache (např. počet přístupů)

Teorie složitosti definuje třídy složitosti pro problémy:

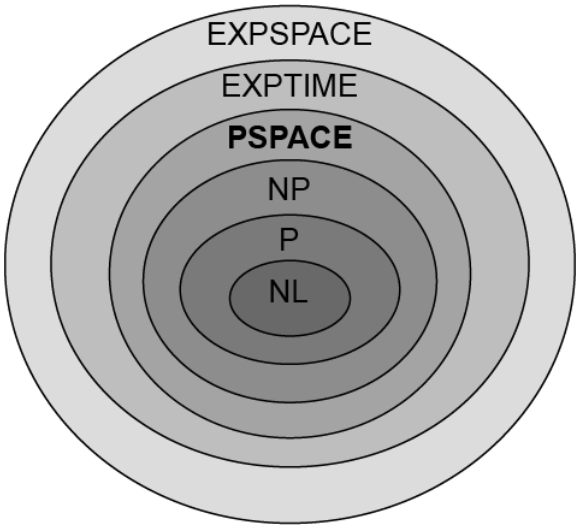
- Třída P
  Rozhodovací úloha U leží ve třídě P právě tehdy, když existuje Turingův stroj, který rozhodne jazyk L_U v polynomiálním čase.
  Př.: Hledání nejkratší cesty, hledání minimální kostry grafu
- Třída NP
  Rozhodovací úloha U leží ve třídě NP právě tehdy, když existuje nedeterministický Turingův stroj , který rozhodne jazyk L_U v polynomiálním čase.
  Př.: K-barevnost (lze daný graf obarvit maximálně k barvami?), Klikovost grafu (Existuje v grafu klika o alespoň k vrcholech?)
- Třída NPC – NP-úplný problém
  Problém je ve třídě NP a zároveň platí, že se na ní polynomiálně redukuje každá úloha z třídy NP. NP-úplné úlohy jsou ty „nejobtížnější“ ze všech NP úloh.
  Př.: Problém obchodního cestujícího, Problém batohu

Třídy PSPACE a NPSPACE

Jazyk *L* je ve třídě *PSPACE* právě tehdy, když existuje deterministický Turingův stroj, který pracuje s polynomiální paměťovou složitostí (tj. nepoužije žádnou paměťovou buňku na indexu vyšším než *p(n)*) a přijímá jazyk *L*.

Jazyk *L* je ve třídě *NPSPACE* právě tehdy, když existuje nedeterministický Turingův stroj, který pracuje s polynomiální paměťovou složitostí a přijímá jazyk *L*.

Bylo dokázáno, že $NP \subseteq NPSPACE$ a dále dle Savitchovy věty platí, že $PSPACE = NPSPACE$.



11. Jazyky a automaty

Jazyky a automaty jsou základním kamenem teoretické informatiky.

Jazykem rozumíme libovolnou neprázdnou množinu *V*, která je *abecedou* a její prvky jsou *znaky* nebo *symboly*.

Definujeme:

- *Slovo* nad abecedou *V* je konečná posloupnost znaků z *V*, $w = a_1 a_2 \dots a_n$
- *Délka slova w* – délka posloupnosti *w*, $|w| = n$
- *Prázdné slovo* ε – posloupnost délky 0
- Množina všech slov nad abecedou V^*
- Množina všech neprázdných slov nad abecedou V^+
- Gramatika G je čtveřice (N, Σ, P, S), kde:
- *N* je konečná množina neterminálních symbolů (neterminálů).
- Σ je konečná množina terminálních symbolů tak, že žádný symbol nepatří do *N* a Σ zároveň.
- *P* je konečná množina odvozovacích pravidel. Každé pravidlo je tvaru $(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$
- *S* je prvek z *N* nazývaný počáteční symbol.

Konvence

- jednotlivé terminály značíme – a, b, c, …
- řetězce terminálů značíme – u, v, w, …
- jednotlivé neterminály – A, B, C … X, Y, Z
- řetězce neterminálů a terminálů – α, β, γ, …
- prázdný řetězec značíme symbolem e nebo také ε

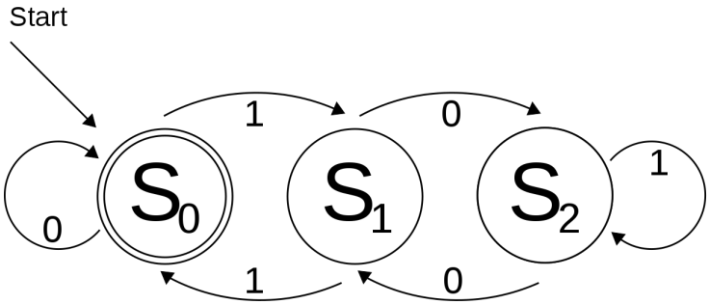
Gramatiky dělíme na několik typů podle Chomského hierarchie.

- Typ 0 – Všechny formální gramatiky (neomezené gramatiky)
- Typ 1 – Kontextové gramatiky
- Typ 2 – Bezkontextové gramatiky
- Typ 3 – Regulární gramatiky

Konečný automat je teoretický výpočetní model používaný v informatice pro studium formálních jazyků. Popisuje velice jednoduchý počítač, který může být v jednom z několika stavů, mezi kterými přechází na základě symbolů, které čte ze vstupu. Množina stavů je konečná (odtud název), konečný automat nemá žádnou další paměť, kromě informace o aktuálním stavu. Rozlišujeme Deterministický KA - v každém místě tabulky má právě jeden cílový stav - a Nedeterministický KA - v každém bodě tabulky není jeden cílový stav, ale celá množina stavů. V přechodové tabulce je také navíc sloupeček pro *prázdný vstup*, označovaný ε. Libovolný nedeterministický automat lze na deterministický převést. Původní množinu stavů je potřeba nahradit její potenční množinou. Každý stav takto vytvořeného automatu pak odpovídá nějaké množině stavů původního nedeterministického automatu a jsou mezi nimi jednoznačné přechody.

Formálně je konečný automat definován jako uspořádaná pětice $(S, \Sigma, \sigma, s, A)$, kde:

- S je konečná neprázdná množina stavů.
- Σ je konečná neprázdná množina vstupních symbolů, nazývaná abeceda.
- σ je tzv. přechodová funkce (též přechodová tabulka), popisující pravidla přechodů mezi stavy. Může mít buď podobu S × Σ → S (deterministický automat), nebo S × {Σ ∪ ε} → P(S) (nedeterministický automat), viz níže.
- s je počáteční stav, s ∈ S.
- A je množina přijímajících stavů, A ⊆ S.



Činnost automatu:

Na počátku se automat nachází v definovaném počátečním stavu. V každém kroku přečte jeden symbol ze vstupu a přejde do stavu, který je dán hodnotou, která v přechodové tabulce odpovídá aktuálnímu stavu a přečtenému symbolu. Pokračuje čtením dalšího symbolu ze vstupu, dalším přechodem podle přechodové tabulky atd. Podle toho, zda automat skončí po přečtení vstupu ve stavu, který patří do množiny přijímajících stavů, platí, že automat buď daný vstup přijal, nebo nepřijal. Množina všech řetězců, které daný automat přijme, tvoří regulární jazyk.

12. Turingovy stroje

Churchova (Church-Turingova) teze: Turingovy stroje (a jim ekvivalentní systémy) definují svou výpočetní silou to, co intuitivně považujeme za efektivně vyčíslitelné. Ke každému algoritmu existuje ekvivalentní Turingův stroj.

Churchovu (Church-Turingovu) tezi nelze formálně dokázat, je však podpořena řadou argumentů:

Turingovy stroje jsou velmi robustní – různé úpravy nemění jejich výpočetní sílu

Byla navržena řada odlišných výpočetních modelů, jejichž síla odpovídá Turingovým strojům

Není znám žádný výpočetní proces, který bychom označili za efektivně vyčíslitelný a který by nebylo možné realizovat Turingovým strojem

Turingův stroj je teoretický model počítače. Skládá se z několika částí::

- Procesorová jednotka – konečný automat
- Program – pravidla přechodové funkce
- Pravostranně nekonečná páska – slouží pro zápis mezivýsledků

Využívá se pro modelování algoritmů v teorii vyčíslitelnosti.

Turingovsky úplné jsou právě ty programovací jazyky a počítače, které mají stejnou výpočetní sílu jako Turingův stroj

Definice:

$M = (Q, \Gamma, b, \Sigma, s, \delta, F)$	Turingův stroj
Q	konečná množina vnitřních stavů
Γ	konečná abeceda symbolů na pásce
$b \in \Gamma$	prázdný symbol, není součástí vstupní abecedy přijímaného řetězce
$\Sigma \subseteq \Gamma \setminus b$	konečná množina vstupních symbolů
$s \in Q$	počáteční stav
$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$	přechodová funkce, L přesun hlavy doleva, R přesun hlavy doprava
$F \subseteq Q$	množina koncových stavů

Konfigurace:	$\langle q, s, n \rangle \in Q \times \{yb^\omega \mid y \in \Gamma^*\} \times N_0$
q	aktuální stav
s	nejmenší souvislá část pásky, obsahující neprázdné symboly
n	pozice čtecí hlavy (číslo buňky)

Pokud Q, Γ jsou disjunktní množiny, lze použít kompaktní tvar

Páska: 1234

Stav stroje: q

Pozice čtecí hlavy: 2

Konfigurace: 1q234

Počáteční konfigurace TS pro vstup $w \in \Gamma^*$

$(s, wb^\omega, 0)$, kompaktní zápis: sw

Postup výpočtu Turingova stroje:

Pokud je aktuální stav stavem koncovým, ukončí výpočet

Čtecí hlava přečte jeden symbol z buňky, na které se právě nachází

Pokud je v přechodové funkci pro aktuální stav a pro přečtený symbol definovaný přechod, provede se (v případě více možných přechodů u nedeterministických strojů se vybere jeden náhodně):

Změní stav

Na aktuální pozici hlavy se zapíše příslušný symbol

Hlava se příslušným způsobem posune (neposune)

Existují nejrůznější modifikace TS.

TS s možností provedení výpočtu bez posunutí hlavy

Přechodová funkce rozšířená o $N - \delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$

Posun N (žádný posun čtecí hlavy) můžeme na běžném TS zařídit tak, že čtecí hlava přečte vstupní symbol z buňky, na které se nachází, provede příslušnou operaci a posune se doprava (R). Nyní opět přečte symbol z buňky, zapíše tam stejný symbol (tj. nezmění symbol, na kterém se hlava nachází) a posune se doleva (L). Tím jsme dostali čtecí hlavu do předchozího umístění a žádný jiný symbol nebyl změněn.

TS s oboustranně nekonečnou páskou

páska není zleva ohraničená

možný posun čtecí hlavy doleva z libovolné konfigurace

n-páskový TS

čte z a zapisuje do více pásek najednou

jediná změna je v přechodové funkci: $\delta: Q \times \Gamma^n \rightarrow Q \times (\Gamma \times \{L, R, N\})^n$

Nedeterministický TS (NTS)

Umožňuje „výběr z více možností“

$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R,N\}}$$

Subrutiny slouží k usnadnění tvorby složitějších TS. Subrutina je množina stavů, která obsahuje počáteční a koncový stav. Zpravidla řeší nějaký dílčí problém v TS. Při běžném programování je ekvivalentem funkce.

Univerzální TS – *U* jako vstup přijímá kód jiného TS *T* a vstupní slovo stroje *T*. Rozkóduje přechodovou funkci stroje *T* a simuluje výpočet tohoto stroje. Dokáže vypočítat libovolnou částečně rekurzivní funkci (neboli je ekvivalentní k univerzální částečně rekurzivní funkci), rozhoduje jakýkoliv rekurzivní jazyk a přijímá libovolný rekurzivně spočetný jazyk.

13. Teorie vyčíslitelnosti

Základní otázka teorie vyčíslitelnosti zní: Co všechno je a co není algoritmicky vyčíslitelné (řešitelné)? Ke kterým problémům existují algoritmy, jež je řeší?

Problém je definován:

- Název: XY
- Instance: co může být vstupem
- Výsledek: výstup, vztah výstupu ke vstupu

Problém je částečné zobrazení typu $\Sigma^* \rightarrow \Sigma^*$, pro platný vstup dává platný výstup, pro neplatný vstup speciální výsledek „Nekorektní zadání".

Problémy mají následující vlastnosti:

- **Algoritmická řešitelnost**
  daném problému řekneme, že je algoritmicky řešitelný (neboli odpovídající (částečné) zobrazení $\Sigma^* \rightarrow \Sigma^*$ je algoritmicky vyčíslitelné), jestliže existuje algoritmus, který je schopen jako vstup přijmout libovolnou instanci daného problému a jeho výpočet pro libovolný takový vstup, vždy skončí, přičemž výstupem bude požadovaný výsledek.
- **Algoritmická rozhodnutelnost**
  daném problému typu Ano/Ne řekneme, že je algoritmicky rozhodnutelný, jestliže existuje algoritmus, který je schopen jako vstup přijmout libovolnou instanci daného problému a jeho výpočet pro libovolný takový vstup vždy skončí, přičemž výstupem bude požadovaná odpověď Ano/Ne.
  Problém je algoritmicky řešitelný, pokud existuje algoritmus, který ke každému konkrétnímu zadání problému vždy v konkrétním čase jednoznačně určí požadovaný výsledek. Rozhodnutelnost je pro problémy typu Ano/Ne
- **Částečná rozhodnutelnost**
  daném problému typu Ano/Ne řekneme, že je částečně rozhodnutelný, jestliže existuje algoritmus, jehož výpočet skončí právě pro takové vstupy, které odpovídají instancím daného problému s odpovědí Ano.
  Problém je částečně rozhodnutelný, pokud známe algoritmus, který se zastaví a vydá výsledek, pouze pokud je odpověď na danou otázku Ano. Pokud je správná odpověď Ne, tak se ji nikdy nedozvíme, protože příslušný algoritmus se nikdy nezastaví.
  Teorie složitosti nám poskytuje zajímavé výsledky. Jedním z nich je tzv. problém zastavení. Nelze zkonstruovat algoritmus, který by pro obecný program ověřil konečnost jeho běhu.
  Teorie složitosti nám poskytuje zajímavé hypotézy. Jednou z nich je Church-Turingova teze. Ke každému algoritmu existuje ekvivalentní Turingův stroj.
- **Problém zastavení**
  Zadání: Znáte-li zdrojový kód programu a jeho vstup, rozhodněte, zda

program zastaví, nebo zda poběží navždy bez zastavení.

V roce 1936 Alan Turing dokázal, že obecný algoritmus, který by řešil problém zastavení pro všechny vstupy všech programů, neexistuje. Problém zastavení se proto označuje jako algoritmicky nerozhodnutelný problém.

Důkaz:

Nerozhodnutelnost problému zastavení lze dokázat sporem.

Počáteční předpoklad: Předpokládejme, že problém zastavení lze rozhodnout. To znamená, že existuje nějaký program Zastaví(program, vstup), který je univerzálním řešením problému zastavení – předpokládáme tedy, že pokud tomuto programu předáme libovolný program a jeho vstup, pak program Zastaví v konečném čase vrátí odpověď takovou, že pokud by volání program(vstup) po konečném počtu kroků skončilo, pak Zastaví(program, vstup) vrátí hodnotu ANO, v opačném případě (pokud by se volání program(vstup) zacyklilo) vrátí hodnotu NE.

Následně zkonstruujme program Paradox(program), který zavolá Zastaví(program, program) a pokud toto volání vrátilo ANO, tak se záměrně zacyklí, a pokud vrátilo NE, tak ihned skončí.

Nyní se ptejme, co je výsledkem volání Paradox(Paradox).

Předpokládejme na chvíli, že Zastaví(Paradox, Paradox) vrací ANO. Z definice programu Paradox pak víme, že se Paradox(Paradox) zacyklí. Podle definice programu Zastaví ale platí, že pokud se Paradox(Paradox) zacyklí, pak musí Zastaví(Paradox, Paradox) vracet NE. Došli jsme ke sporu.

Předpokládejme na druhou stranu na chvíli, že Zastaví(Paradox, Paradox) vrací NE. Z definice programu Paradox pak víme, že se Paradox(Paradox) zastaví. Podle definice programu Zastaví ale platí, že pokud se Paradox(Paradox) zastaví, pak musí Zastaví(Paradox, Paradox) vracet ANO. Opět jsme došli ke sporu.

Protože jsme vyčerpali všechny možnosti a vždy došli ke sporu, musí být nepravdivý počáteční předpoklad. Z toho vyplývá, že program Zastaví(program, vstup), tak jak byl definován, neexistuje.

Churchova–Turingova teze

Hypotéza říká, že každý možný výpočet lze úspěšně uskutečnit algoritmem běžícím na počítači, je-li k dispozici dostatek času a paměti.

Algoritmus musí splňovat následující požadavky:

- Algoritmus se skládá z konečného počtu instrukcí, které jsou přesně definovány použitím konečného počtu symbolů.
- Algoritmus vždy vrátí výsledek po konečném počtu kroků.
- Algoritmus může provádět i člověk s tužkou a papírem.

- Spuštění algoritmu nepotřebuje lidskou inteligenci, s výjimkou té, která je třeba na pochopení a vykonání instrukcí.

Jelikož každý počítačový program lze přeložit do jazyka Turingova stroje a také naopak, lze tezi ekvivalentně formulovat pro kterýkoli běžně používaný programovací jazyk.

Programovací jazyk potřebuje jednu z následujících konstrukcí (kromě jiných), aby byl turingovsky úplný (tj. ekvivalentní Turingovu stroji):

- cyklus while-do,
- neomezenou (alespoň teoreticky) rekurzi,
- podmíněný skok.

Běžné programovací jazyky mívají všechny tři tyto konstrukce. Mezi jazyky, které turingovsky úplné nejsou, patří SQL (myšleno bez uložených procedur).

ALGORITMY A DATOVÉ STRUKTURY

1. Algoritmus

Pojmem algoritmus se myslí přesný návod nebo postup, kterým lze vyřešit daný typ úlohy. Popisuje teoretický princip řešení, na rozdíl od přesného (konkrétního) zápisu v daném programovacím jazyce.

Algoritmus by měl splňovat jisté vlastnosti:

Konečnost – Každý algoritmus musí končit v konečném počtu kroků. Tento počet kroků může být libovolně velký, ale pro každý jednotlivý vstup musí být konečný. Postup, který tuto podmínku nesplňuje, nelze nazývat algoritmem, ale jen výpočetní metodou.

Obecnost – Algoritmus neřeší jeden konkrétní problém, ale obecnou třídu obdobných problémů.

Determinovanost – Každý krok algoritmu musí být jednoznačně a přesně definován. V každé situaci musí být zřejmé, co a jak se má provést, jak má algoritmus pokračovat. Některé algoritmy nejsou determinované – obsahují prvek náhody (např.: genetické algoritmy)

Výstup (neboli resultativnost) – algoritmus má alespoň jeden výstup, veličinu, která je v požadovaném vztahu ke vstupům. Výstup tvoří odpověď na problém.

Elementárnost – algoritmus se skládá z konečného počtu jednoduchých (elementárních) kroků

1.1. Proces vývoje algoritmů

Existuje několik postupů, které slouží k navrhování algoritmů:

Metoda shora dolů – postup řešení rozkládáme na jednodušší operace, dokud nedospějeme k elementárním krokům.

Metoda zdola nahoru – z elementárních kroků vytváříme prostředky, které nakonec umožní zvládnout požadovaný problém.

Případně, **kombinace obou uvedených metod**, kdy postup shora dolů doplníme částečným krokem zdola nahoru. Například použijeme knihovnu funkcí, vyšší programovací jazyk nebo systém pro vytváření programů.

Obvyklými postupy algoritmů jsou metody: Rozděl a panuj, Hladový algoritmus, Dynamické programování a backtracking.

- **Metoda Rozděl a panuj**, dělí problém na dílčí úlohy, které musí být na sobě nezávislé. Tyto dílčí úlohy poté řeší. Často je implementován rekurzivně nebo

iteračně.

- **Haldový algoritmus** se většinou používá pro řešení optimalizačních úloh. Vybírá vždy lokální minimum (maxima) ve snaze najít globální minimu (maximum). Díky tomuto postupu není vždy ideální a nemusí dosáhnout globálního minima (maxima)
- **Dynamické programování** dělí problém na dílčí úlohy, obdobně jako metoda Rozděl a panuj, ale v tomto případě mohou být tyto části závislé.
- **Backtracking**, neboli hledání s návratem je způsob řešení algoritmických problémů založený na prohledávání stavového stromu problému. Je to vylepšení řešení hrubou silou (brute-force)

1.2. Typy algoritmů

Algoritmy lze rozdělit na několik druhů:

- **Rekurzivní algoritmy**, které využívají (volají) samy sebe.
- **Pravděpodobnostní (probabilistické),** které provádějí některá rozhodnutí náhodné (pseudonáhodné) volby.
- **Paralelní algoritmy**, které dělí úlohy mezi více počítačů (procesorů, vláken).
- Dalším typem jsou **genetické algoritmy** pracující na základě napodobení biologických evolučních procesů.
- A **heuristické algoritmy**, které nehledají přesné konkrétní řešení, ale jen nějaké vhodné přiblížení. Používají se v situacích, kdy dostupné zdroje nepostačují na využití exaktních algoritmů, nebo nejsou žádné vhodné exaktní algoritmy vůbec známy.

2. Abstraktní datový typ – ADT

2.1. Popis

Abstraktní datový typ – ADT – je výraz používající se na pro typy dat, která jsou nezávislá na vlastní implementaci.

Používáním ADT se snažíme o zjednodušení a zpřehlednění programu, který provádí operace s daným datovým typem.

Každý ADT je realizovatelný pomocí základních algoritmických operací, jako je přiřazení, sčítání, násobení, podmíněný skok atd.

2.1. Vlastnosti ADT

ADT má mít následující vlastnosti:

- **Všeobecnost implementace** – jednou navržený ADT může být zabudován a bez problémů používán v jakémkoliv programu.
- **Přesný popis** – propojení mezi implementací a rozhraním musí být jednoznačné a úplné.
- **Jednoduchost** – uživatel se nemusí starat o vnitřní realizaci a správu ADT v paměti.
- **Zapouzdření** – rozhraní jako uzavřená část, uživatel ví, co ADT dělá, ale ne jak to dělá
- **Integrita** – uživatel nemůže zasahovat do vnitřní struktury dat
- **Modularita** – „stavebnicový" princip programování je přehledný a umožňuje snadnou výměnu části kódu. Při hledání chyb mohou být jednotlivé moduly považovány za kompaktní celky. Při zlepšování ADT není nutné zasahovat do celého programu.

Pokud je ADT programován objektově, pak jsou většinou tyto vlastnosti implicitně splněny.

2.1. Operace v ADT

S ADT lze provádět základní typy operací. Mezi ně patří Konstruktor, Selektor a Modifikátor.

- **Konstruktor** vytváří novou hodnotu ADT, stará se o sestavení platné vnitřní reprezentace hodnoty na základě dodaných parametrů.
- **Selektor** má na starosti získání hodnot, které tvoří složky nebo vlastnosti konkrétní hodnoty ADT.
- A **modifikátor** provádí změny hodnot datového typu.

3. Analýza algoritmů

3.1. Porovnání efektivity procesů

Jelikož stejný problém lze obvykle řešit několika různými postupy (algoritmy), je proto potřeba mít nějaký nástroj, který nám umožní porovnávat efektivnost daného postupu. Buď můžeme algoritmy porovnávat **experimentálně**, nebo **teoreticky**.

Experimentální analýze je časově náročná. Časová náročnost se samozřejmě zvyšuje s množstvím a velikostí vstupů. Tento typ analýzy vyžaduje konkrétní implementaci algoritmu, což samozřejmě sebou nese nároky na další znalosti. Těžko se hledá nějaký průměrný případ. Proto je lepší soustředit se na nejhorší možný případ, který se lehce analyzuje a je kritický pro většinu aplikací, ať už se jedná o hry, finance, robotiku, automatické operace.

Experimentální analýza časové náročnosti probíhá v prostředí, kde běží daná implementace algoritmu (program), většinou za pomoci nějaké interní funkce pro měření času. Běh programu závisí na vstupech a jejich složení, navíc ne všechny vstupy jsou zahrnuty v každém běhu programu. Porovnání dvou algoritmů vyžaduje stejný hardware i software a stejné obsazení paměti,

Místo experimentální analýzy lze použít jisté **teoretické postupy**. Teoretická analýza využívá popis algoritmu pomocí operací namísto konkrétní implementace. Bere do úvahy všechny vstupy a umožňuje ohodnotit rychlost algoritmu nezávisle na hardware nebo software.

Jedním z těchto nástrojů je tzv. **Pseudo-kód**. Pseudo-kód umožňuje a využívá vyšší úrovně popisu algoritmů. Popis je více strukturovaný než běžně psaný text, ale méně detailní než konkrétní implementace. Je to preferovaný zápis pro popis algoritmů. Jeho výhodou i nevýhodou je, že skrývá problémy konkrétní implementace.

Pseudo-kód využívá klíčová slova pro popis algoritmu:

Pro řízení běhu: -If...then...else (condition), while...do, repeat...until, for...do (cycles)

Hlavička: Algorithmus `Name(arg1, arg2...)`, input, output

Volání procedury (Methoden, Algorithmus): `var.Name(arg1,arg2,...)`

Návrat hodnoty: `return` *`Expression`*

Výrazy:	←	Zuschreibung
	=	Gleichheit
	+, -, n^2,...	Mathematische Operationen

3.1. Primitivní operace

Primitivní operace je základní operace provedená algoritmem, je identifikovatelná v pseudo-kódu, nezávislá na programovacím jazyku a měla by být přesně definovaná. Takovou primitivní operací může být třeba vyhodnocení výrazu, přiřazení hodnoty do proměnné, indexování v poli, volání nebo návrat z procedury a podobně.

Podobně můžeme využít **asymptotickou notaci (big O, Bachmann-Landau notace).** Určuje operační náročnost algoritmu tak, že zjišťuje, jakým způsobem se bude chování algoritmu měnit v závislosti na změně velikosti (počtu) vstupních dat. Obvykle se používá asymptotická časová a prostorová složitost. Používaný zápis znamená, že náročnost algoritmu je menší než $A+B \cdot f(N)$, kde A a B jsou vhodně zvolené konstanty a N je veličina popisující velikost vstupních dat. Zanedbáváme tedy multiplikativní i aditivní konstanty, tzn. $O(N+1000)=O(1000 \cdot N)=O(N)$. Zajímá nás jen chování funkce pro velké hodnoty N.

Pro určení časové náročnosti algoritmu pomocí big O notace musíme nalézt největší možný počet primitivních operací, který pak vyjádříme pomocí big O notace.

4. Fronty a zásobníky

4.1. Zásobník

Zásobník je datová struktura, která slouží pro ukládání dat. Je charakterizován způsobem manipulace s daty – k datům přistupuje pomocí principu LIFO (**Last In First Out**). Lze si ho představit jako zásobník na talíře.

ADT zásobník musí obsahovat minimálně operace pro:

- vložení objektu,
- vrácení a odebrání posledního objektu,
- dotaz na vrchol zásobníku,
- jeho velikost,
- jestli je zásobník prázdný.

Pokud se pokusíme provést na prázdném zásobníku operaci pop, nebo top, dostaneme výjimku *EmptyStackException*.

Aplikací zásobníku je například historie prohlížeče webových stránek, Undo sekvence v editorech nebo řetězec volání jednotlivých procedur. Zásobník lze využít jako pomocnou datovou strukturu pro jiné algoritmy, případně jako část jiných datových struktur.

Zásobník lze implementovat nejjednodušeji pomocí pole. Prvky přidáváme zleva doprava a pomocná proměnná drží index posledního prvku.

Díky vlastnostem pole získáme následující vlastnosti:

- n – počet prvků v zásobníku
- Paměťová náročnost - $O(n)$
- Časová náročnost každé operace – $O(1)$

Pole nám však přináší také jistá omezení:

- Na začátku musíme definovat velikost zásobníku
- Velikost zásobníku nelze jednoduše změnit
- Přidání prvku do plného zásobníku vyvolá výjimku specifickou pro implementaci

4.2. Fronta

Fronta je datová struktura typu **FIFO** (**First In First Out**).

Minimální implementace fronty musí obsahovat operace pro:

- vložení položky na konec fronty,
- výběr položky ze začátku fronty,
- dotaz na začátek fronty,
- její délky a obsazenost.

Stejně jako zásobník i fronta může vyhodit výjimku při operaci *dequeue* nebo front nad prázdnou frontou – *EmptyStackException*.

Aplikací fronty je například pořadník, fronta, přístup ke sdíleným zdrojům (tiskárny), multiprogramování. Frontu lze využít také jako pomocnou datovou strukturu pro jiné algoritmy, případně jako část jiných datových struktur.

Frontu lze implementovat pomocí pole. Pro zlepšení vlastností se využívá kruhového pole. Máme pak dvě proměnné, *f* – index prvního prvku a *r* index posledního prvku zvětšený o jedna (ukazuje na první volné místo).

5. Vektory, Seznamy, Sekvence

5.1. Vektory

Vektor rozšiřuje pojem pole ukládáním sekvence libovolných objektů. Prvek ve vektoru může být čten, vkládán a odebírán pomocí určení jeho pořadí.

Vektor umožňuje **základní operace**:

- prvek na určitém pořadí,
- záměnu prvku na konkrétním místě,
- vložení na konkrétní místo a odebrání prvku z konkrétního místa,
- zjistit velikost, a jestli je daný vektor prázdný.

Operace s vektory mohou vyhodit výjimku v případě nesprávného indexu (obvykle záporného). Aplikací vektoru je tříděná kolekce objektů (základní databáze).

Vektor lze implementovat pomocí pole. To nám pak přináší následující vlastnosti:

- Proměnná ***n*** určuje délku vektoru
- Operace *isEmpty(), elemAtRank(r), replaceAtRank(r, O)* - časová náročnost *O(1)*
- Operace *insertAtRank(r, O)* - časová náročnost *O(n)*
- Operace *removeAtRank(r)* - časová náročnost *O(n)*

5.2. Seznamy

Další datovou strukturou je **Seznam**. Seznam je posloupnost pozic ukládající libovolná data. Zavádí vztahy před/po mezi pozicemi.

Obecnými operacemi jsou dotaz na velikost, a dotaz na prázdný seznam. Další operace jsou dotaz, jestli je daný prvek první nebo poslední, získání prvního a posledního prvku, dále pak prvek předcházejí, či následující.

ADT seznam obsahuje operace:

- replaceElement(p, o),
- swapElements(p, q),
- insertBefore(p, o),
- insertAfter(p, o),
- insertFirst(o),
- insertLast(o),

- remove(p).

Seznamy lze rozdělit na single linked list – jednosměrný spojový seznam - a double linked list – obousměrný spojový seznam.

V jednosměrném seznamu prvek obsahuje odkaz na následující uzel, v případě obousměrného seznamu prvek obsahuje odkaz i na předcházející uzel.

5.3. Sekvence

ADT **Sekvence** je spojením vektoru a seznamu, k prvkům tak lze přistupovat jak pomocí pozice, tak i pořadí. Kromě vektorových a seznamových operací obsahuje i propojující operace **atRank(r)** a **rankOf(p).**

Sekvence je obecný základní typ použitelný pro ukládání uspořádaného souboru prvků. Je obecnou náhradou za zásobník, frontu, vektor nebo seznam. Lze ji použít i jako malou databázi.

6. Stromy

Strom představuje model hierarchické struktury, která se skládá z uzlů, mezi nimiž je vztah rodič-dítě.

Strom lze použít jako organizační diagram, souborové systémy, či programovací prostředí.

Pro popis stromů a jejich částí se využívá následující terminologie:

6.1. Druhy uzlů

- kořen (root),
- vnitřní uzel (inner node) - uzel, který není kořenem, ani listem,
- list (leaf node, external node) - uzel, který nemá žádné potomky,
- rodič (parent node) - uzel, který přímo předchází daný uzel na cestě od listu ke kořeni,
- potomek (child node) - uzel, který přímo následuje za daným uzlem na cestě od kořene k listu,
- sourozenec (sibling) - jako sourozenci se označují uzly se stejným rodičem,
- předek (ancestor node, predecessor node) - uzel, který leží před daným uzlem na cestě ke kořeni (nejbližší předek je rodič),
- následník (successor node) - uzel, který leží za daným uzlem na cestě od kořene k libovolnému listu (nejbližší následník je potomek),
- hloubka (depth) - hloubka stromu je délka nejdelší cesty od kořene k listu, přičemž prázdný strom má definovánu hloubku jako -1,
- úroveň (level) - většinou se používá ve významu množiny uzlů, které se nachází ve stejné vzdálenosti od kořene, počítáno dle počtu uzlů.

6.2. Struktura

- podstrom (subtree) - podgraf stromu, který je také stromem (obecně se nejčastěji setkáváme s podstromy tvořenými tak, že se vezme nějaký uzel stromu jako nový kořen a zbytek struktury se zachová),
- větev (branch) - každá cesta od kořene k listu.

6.3. Operace pro manipulaci se stromy

Obecné operace:

- integer size(),
- boolean isEmpty(),
- objectIterator elements(),
- positionIterator position().

Přístupové operace

- position root(),
- position parent(),
- positionIterator children(p),
- Dotazovací operace,
- boolean isInternal(p),
- boolean isExternal(p),
- boolean isRoot(p).

Aktualizační operace

- swapElements(p, q),
- object replaceElement(p, o).

Jelikož je strom hierarchickou strukturou, lze jej procházet několika způsoby.

Pre-order průchod

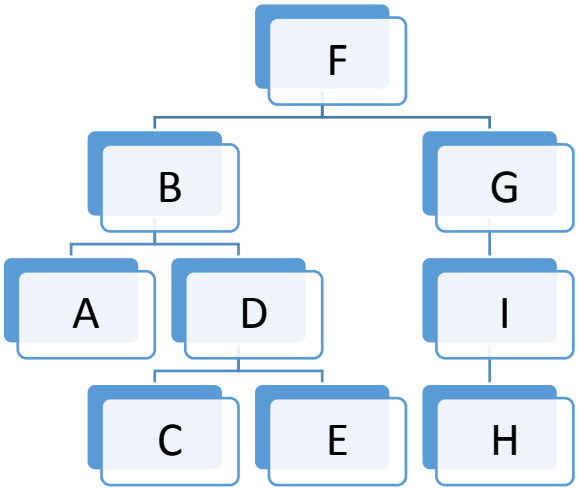
- Kontrola jestli je uzel prázdný nebo null,
- Zobrazení dat aktuálního uzlu,
- Průchod levým podstromem rekurzivním voláním pre-order funkce,
- Průchod pravým podstromem rekurzivním voláním pre-order funkce.

In-order průchod

- Kontrola jestli je uzel prázdný nebo null,
- Průchod levým podstromem rekurzivním voláním in-order funkce,
- Zobrazení dat aktuálního uzlu,
- Průchod pravým podstromem rekurzivním voláním in-order funkce.

Post-order průchod

- Kontrola jestli je uzel prázdný nebo null,
- Průchod levým podstromem rekurzivním voláním post-order funkce,
- Průchod pravým podstromem rekurzivním voláním post-order funkce,
- Zobrazení dat aktuálního uzlu.



Každý typ průchodu nám poskytuje jiné výsledky.

- Pre-order: F, B, A, D, C, E, G, I, H
- In-order: A, B, C, D, E, F, G, H, I
- Post-order: A, C, E, D, B, H, I, G, F

Pomocí ADT strom lze definovat i Binární strom, či jiné další typy.

Binární strom rozšiřuje definici stromu o to, že každý uzel má nejvýše dvě děti, které tvoří uspořádanou dvojici (levý potomek, pravý potomek).

Binární strom přidává další operace:

- position leftChild(p)
- position rightChild(p)
- position sibling(p)

7. Prioritní fronta a Halda

7.1. Prioritní fronta

Prioritní fronta uchovává kolekci položek, kde položka je uspořádanou dvojicí klíč (priorita)-hodnota.

Základní implementace by měla obsahovat operace:

- **insertItem(k, o),**
- **removeMin(),**
- **minKey(k, o),**
- **minElement(),**
- **size(),**
- **isEmpty().**

Aplikací prioritní fronty jsou např.: aukce a burzy.

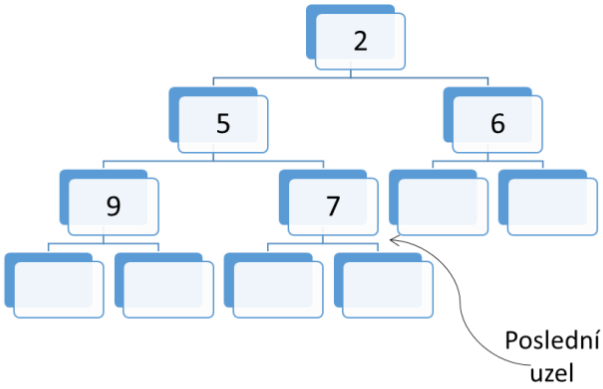
Klíčem prioritní fronty mohou být libovolné objekty, na nichž lze definovat pořadí a lze je uspořádat. Dva rozdílné prvky (hodnoty) mohou mít stejný klíč (prioritu).

Pro použití prioritní fronty je nutné zavést ADT Comparator, který nám umožní porovnávat dva objekty.

7.2. Halda (Heap)

Halda (Heap) je binární strom, který uchovává klíče jako interní uzly. Pro každý uzel stromu mimo kořen platí, že klíč uzlu je větší než klíč jeho rodiče. Pro haldu definujeme kompletní binární strom. Nechť h je výška stromu, pak pro i do 0 do $h-1$ je 2^i uzlů hloubky i.

Poslední uzel haldy, je vnitřní uzel, který se nachází nejvíce vpravo na úrovni $h-1$.



Haldu lze použít pro implementaci prioritní fronty. Pak ukládáme položku (klíč, hodnota) v každém interním uzlu a uchováváme odkaz na pozici posledního prvku.

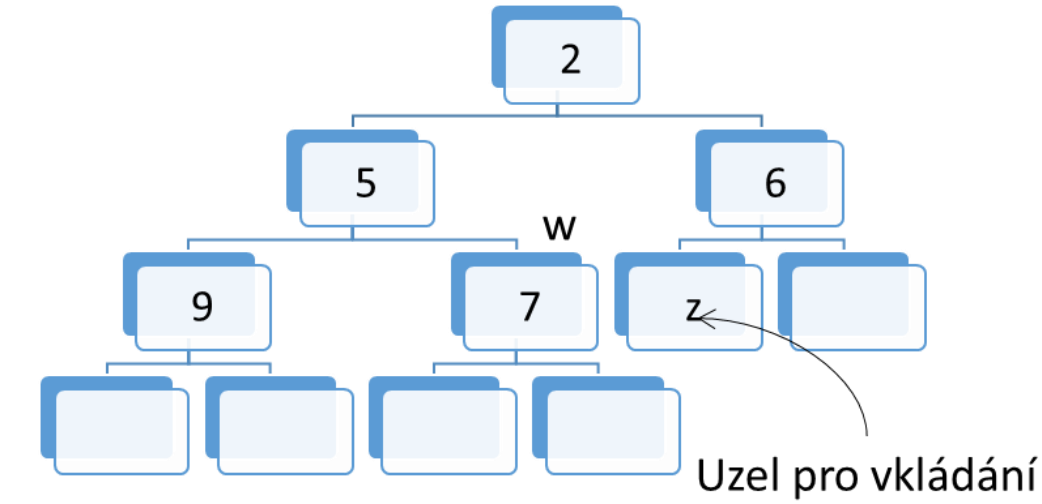
7.3. Manipulace s haldou:

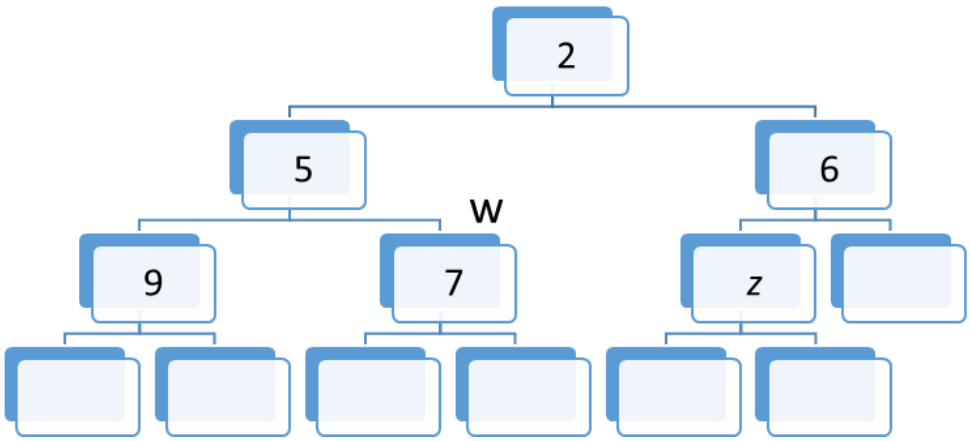
Operace insertItem(k, o)

- Nalezení uzlu, kam se bude vkládat
- Uložení klíče k do uzlu z, změna uzlu z na interní uzel
- Obnovení uspořádanosti haldy (kontrola vlastností) – operace upheap()

Operace removeMin()

- Odpovídá odebrání kořene z hromady (uzel 2)
- Výměna kořene za poslední uzel (2 – 7)
- Změna uzlu w a jeho dětí na list
- Obnovení uspořádanosti haldy – operace downheap()





upheap()

- Vložení uzlu může narušit uspořádání.
- Algoritmus *upheap* obnoví uspořádání prohazováním klíče *k* vzhůru od vloženého uzlu.
- Končí ve chvíli, kdy se vložený uzel stane kořenem, nebo když rodičovský klíč je roven nebo menší než *k.*

downheap()

- Odebrání kořene může narušit uspořádání.
- Algoritmus *downheap* obnoví uspořádání prohazováním klíče *k* dolů od kořene.
- Končí ve chvíli, kdy se vložený uzel stane listem, nebo když klíč potomka je roven nebo větší než *k.*

8. Slovníky a Hashovací tabulky

8.1. Slovník

Jako **Slovník** označujeme ADT obsahující kolekci klíč-hodnota, ve které lze vyhledávat. Operace, které lze se slovníkem provádět:

- **findElement(k),**
- **insertItem(k, o),**
- **removeElement(k),**
- **size(),**
- **isEmpty(),**
- **keys(),**
- **elements()**

Aplikací slovníku je adresář, autorizace kreditních karet, slovník, překlad doménového jména na IP adresu.

Log File – slovník implementovaný jako neuspořádaná sekvence (double linked list). Objekty ukládáme v libovolném pořadí.

Náročnost operací:

- Vložení objektu O(1)
- Nalezení prvku, odebrání prvku O(n)

Log file je vhodný pro malé slovníky, nebo aplikace, kde je vkládání nejčastější operací, zatímco vyhledávání a odebírání se provádí zřídka.

Operace **findElement(k)** na slovníku implementovaném pomocí sekvence založené na poli uspořádaném podle klíčů, se provádí jako binárním vyhledávání. V každém kroku je číslo kandidáta dělené dvěma, končí po logaritmickém počtu kroků.

Vyhledávací tabulka je Slovník implementovaný pomocí uspořádané sekvence. Uchováváme položky slovníku v sekvenci založené na poli uspořádané podle klíčů, je nutný externí comparator pro klíče.

Náročnost operací:

- Nalezení prvku O(log(n))
- Vložení prvku, odebrání prvku O(n)

Efektivní pro malé slovníky, nebo aplikace, kde se nejčastěji provádí vyhledávání.

Binární vyhledávací strom je binární strom, pro který platí:

- u, v a w jsou tři uzly takové, že u je v levém podstromu v a w je v pravém

podstromu v

- $key(u) \leq key(v) \leq key(w)$
- Externí uzly neuchovávají žádnou položku
- In-order průchod dává klíče v zvyšujícím se pořadí.

8.2. Hashovací tabulka

Hash funkce *h*, je taková funkce, která přiřazuje klíči daného typu celočíselnou hodnotu z intervalu od 0 do N-1. Cílem této funkce je uniformě rozdělit klíče v daném intervalu.

Hashovací tabulka pro daný typ klíče obsahuje hash funkci a pole (tabulku) o velikosti N.

Klíč se nahradí hash hodnotou. Může se však stát, že pro dva klíče se vygeneruje stejná hash hodnota – dojde ke kolizi. To lze řešit v zásadě dvěma způsoby:

- zřetězení (chaining) – kdy se kolidující položky ukládají jako sekvence,
- otevřené adresování – položka se uloží na jiné místo v tabulce.

9. Řadící algoritmy I.

9.1. Vysvětlení

Řadící algoritmy, někdy nesprávně označované jako třídící algoritmy, slouží k seřazení daného souboru dat do specifického pořadí, ať už abecedně nebo podle čísel. Dvojice klíč-hodnota se řadí podle klíče a na hodnotu není brán zřetel.

Řadící algoritmy lze rozdělit na stabilní a nestabilní, podle toho, zda zachovávají pořadí položek se stejným klíčem, přirozené a nepřirozené – přirozený pracuje rychleji s částečně seřazenou množinou.

Lze je také rozdělit podle typu řazení:

- Výběrem
- Vkládáním
- Záměnou
- Slučováním

Nejznámější algoritmy – typ algoritmu je obsažený v jeho pojmenování

- Bubble sort Bublinkové řazení
- Heap sort Řazení haldou
- Insertion sort Řazení vkládáním
- Merge sort Řazení slučováním
- Quicksort Rychlé řazení
- Selection sort Řazení výběrem

Další algoritmy založené na jiném principu

- Bucket sort Přihrádkové řazení
- Radix sort Třídění podle základu
- Counting sort Řazení počítáním četnosti

9.2. Bubble sort

- Implementačně jednoduchý algoritmus
- Opakovaně prochází seznam a porovnává dva sousední prvky
- Univerzální, pracuje lokálně (nepotřebuje dodatečnou paměť)
- Prvky s nejvyšší hodnotou probublávají na konec seznamu

Algoritmus:

```
procedure bubbleSort( A : list sortable items )
    n = length(A)
    repeat
        swapped = false
        for i from 1 to n-1 inclusive do
     if A[i-1] > A[i] then
                        swap( A[i-1], A[i] )
                swapped = true
            end if
        end for
    until not swapped
end procedure
```

9.3. Heap sort

- Jeden z nejlepších obecných algoritmů založených na porovnávání prvků
- Není stabilní
- Využívá datovou strukturu halda a její vlastnosti

Algoritmus:

```
procedure heapsort(a, count) is
    input: an unordered array a of length count
    heapify(a, count)
   end ← count - 1
    while end > 0 do
         swap(a[end], a[0])
        (the heap size is reduced by one)
        end ← end - 1
        (the swap ruined the heap property, so restore it)
        shiftDown(a, 0, end)
```

Nejmenší prvek je kořenem – umístíme na první místo v poli a kořen odebereme

downheap() – obnovení haldy podle pravidel. Opakujeme odebírání kořene a obnovení haldydokud halda není prázdná

9.4. Insertion sort

- Jednoduchá implementace
- Efektivní na malých množinách
- Efektivní na částečně seřazených množinách
- Stabilní
- Dokáže řadit data tak, jak přicházejí na vstup
- Postup:
    - Posloupnost rozdělíme na seřazenou a neseřazenou tak, že seřazená obsahuje první prvek posloupnosti.
    - Z neseřazené části vybereme první prvek a zařadíme jej na správné místo v seřazené posloupnosti.
    - Prvky v seřazené posloupnosti posuneme o jednu pozici doprava.
    - Seřazenou část zvětšíme o jeden prvek. Naopak neseřazenou část o jeden prvek zleva zmenšíme.

- Kroky 2–5 aplikujeme až do úplného seřazení neseřazené části.

Algoritmus

```
    for i = 1 to length(A)
        j ← i
        while j > 0 and A[j-1] > A[j]
            swap A[j] and A[j-1]
            j ← j - 1
        end while
    end for
```

9.5. Merge sort

- metoda Rozděl a panuj
- Nejhorší i průměrná časová složitost O(NlogN)
- Potřebuje navíc pole o velikosti N
- Stabilní, paralelizovatelný
- Postup:
    - Rozdělí neseřazenou množinu dat na dvě podmnožiny o přibližně stejné velikosti.
    - Seřadí obě podmnožiny.
    - Spojí seřazené podmnožiny do jedné seřazené množiny.

Algoritmus

```
mergesort(m)
      var list left, right
      if length(m) ≤ 1
            return m
      else
            middle = length(m) / 2
            for each x in m up to middle
                  add x to left
            for each x in m after middle
                  add x to right
left = mergesort(left)
            right = mergesort(right)
            result = merge(left, right)
      return result
```

9.6. Quicksort

- Jeden z nejrychlejších běžných algoritmů řazení založený na porovnávání prvků
- Časová složitost O(NlogN) – O(N2)
- Metoda Rozděl a panuj
- Rekurzivní algoritmus

- Postup:
  - výběr pivot – rozdělení posloupnosti na dvě části – větší a menší než pivot
  - Seřaď obě části stejným způsobem
- Volba pivotu – ideální medián
  - První prvek (jakákoli fixní pozice) – nevýhodné na částečně seřazených množinách
  - Náhodný prvek – ve skutečnosti pseudonáhodný
  - Medián tří (pěti…) – či jiného počtu prvků z fixních nebo náhodných pozic

Při správné implementaci nepotřebuje paměť navíc. Quicksort je nestabilní algoritmus. Způsob volby pivotu má vliv na řazení, ale v průměru jde o nejrychlejší známý univerzální algoritmus pro řazení polí v operační paměti počítače.

9.7. Selection sort

Jednoduchý algoritmus, jehož časová složitost je $O(N^2)$, je vhodný pro malé množství dat. Je univerzální, lokální a nestabilní.

Postup:

- Rozdělíme si posloupnost na seřazenou a neseřazenou část
- Najdeme prvek s nejmenší hodnotou v neseřazené části posloupnosti
- Zaměníme ho s prvkem na první pozici neseřazené části
- První prvek neseřazené části zahrneme do seřazené části a zároveň neseřazenou část zmenšíme o 1 prvek zleva
- Zbytek posloupnosti se uspořádá opakováním kroků 2 až 5 pro zbylou neseřazenou část

Porovnání řadících algoritmů:

Název		Časová složitost			Dodatečná paměť	Stabilní	Přirozená	Metoda
Anglicky	Česky	Minimum	Průměrně	Maximum				
Bubble sort	Bublinkové řazení	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	ano	ano	záměna
Heapsort	Řazení haldou	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$	ne	ne	halda, záměna
Insertion sort	Řazení vkládáním	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	ano	ano	vkládání
Merge sort	Řazení slučováním	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(\log n)$	ano	ano	slučování
Quicksort	Rychlé řazení	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$	ne	ne	záměna
Selection sort	Řazení výběrem	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	zprav. ne	ne	výběr

9.8. Bucket sort

Rozděluje data do několika přihrádek, jeho časová náročnost je O(n*k), kde k=n/m, vstupní data n, počet přihrádek m.

Pro použití Bucket sortu jsou nutné předpoklady:

- Vhodný pro rovnoměrně rozložené hodnoty vstupních dat.
- Algoritmus pro seřazení přihrádek musí být stabilní

Postup:

- Vstupní data jsou rozdělena do předem definovaného počtu přihrádek.
- Na každou přihrádku volán stabilní řadicí algoritmus.
- Jednotlivé přihrádky postupně kopírovány do výstupního pole.

Výhody bucket sortu jsou, že je dobře paralelizovatelný a nemusí mít všechna data v paměti najednou.

9.9. Radix sort

Řadí celá čísla procházením všech číslic. Existují dva přístupy:

- LSD (Least Significant Digit) – řazení podle nejméně významných číslic (odzadu)
- MSD (Most Significant Digit) – nejvíce významné číslice (odpředu)

Časová složitost: O((z+n)*$\log_z u$), kde z je základ zvolené číselné soustavy, *n* počet čísel na vstupu a *u* je maximální rozmezí čísel na vstupu

Nehodí se pro neomezeně velké vstupy.

Příklad LSD radix: 170, 45, 75, 90, 802, 2, 24, 66 ⇒ 170, 90, 802, 2, 24, 45, 75, 66 ⇒ 802, 2, 24, 45, 66, 170, 75, 90 ⇒2, 24, 45, 66, 75, 90, 170, 802

9.10. Counting sort

Vhodný pro velké soubory s malým množstvím diskrétních hodnot, je stabilní. Jeho časová náročnost je *O(N+M)* a paměťová náročnost *O(M.)*

Předpoklady:

- Počet různých hodnot (M) významně menší než celkový počet prvků (N).
- Pomocné pole – zápis a čtení v konstantním čase (pole indexované hodnotou nebo hashem hodnoty)

Algoritmus:

- zleva (či zprava) projde vstupní pole
- pro každý prvek zvýší v pomocném poli četnost výskytu tohoto prvku
- ke každé položce přičte počet výskytů všech předchozích položek (získá přesnou pozici hranice)
- začne zprava procházet neseřazené pole
- pro každý prvek se podívá do pomocného pole na horní hranici pro umístění
- na tuto hranici ho umístí a zároveň ji sníží o jedna
- takto postupuje, dokud neprojde celé pole

10. Pattern matching

Pattern matching, neboli porovnávání vzorů, je vyhledávání jistého vzoru v dané sekvenci. Obvykle hledání podřetězce v řetězci.

Nejprve je nutné si říct, co je to řetězec. Řetězec (string) je sekvence znaků z dané abecedy. Abeceda je množina všech možných znaků – Ascii, Unicode, {0,1}, {A, C, G, T}

Pokud je *P* řetězec délky *m*, pak podřetězec *P[i..j]* řetězce *P* se skládá ze znaků mezi *i* a *j*. Řetězec nacházející se před indexem *i* je předpona – prefix. Řetězec nacházející se za indexem *j* je přípona – suffix.

Aplikace – textové editory, vyhledávací nástroje, biologický výzkum.

Pro pattern matching existuje několik algoritmů.

Základním z nich je **Brute-Force** (hrubá síla).

Prochází text zleva doprava

Porovnává vzor *P* s textem *T*, pro všechny možné pozice dokud:

- Není nalezena shoda
- Nebyly vyzkoušeny všechny možné pozice

Časová náročnost toho algoritmu je *O(nm).*

Boyer-Moorův algoritmus prochází text od konce (zprava doleva).

Definujeme: Index *i* ukazuje do textu *T*, index *j* ukazuje do *P*

V průběhu vyhledávání mohou nastat 4 případy:

- T(i) není v P vůbec, posuneme i dále o délku P (zarovnáme P na další písmeno v T, tedy T(i+1))
- T(i) odpovídá P(j) - posuneme se v obou do leva a opakujeme (jako v brutální síle)
- T(i) není P(j), ale T(i) je v P před indexem j -> zarovnáme P doprava tak, aby T(i) odpovídal jeho výskytu v P a opakujeme
- T(i) není P(j), ale T(i) je v P za indexem j -> posuneme se doprava o 1 a opakujeme (nemůžeme se vracet, ale ani posunout o více dále)

Vrátíme *i* - pokud nalezneme celý pattern

Algoritmus je rychlejší než Brute-Force, přesto jeho složitost může být *O(mn + A),* kde A je velikost abecedy.

Pro aplikaci tohoto algoritmu je nutný preprocessing:

- zjistí, na které pozici má které písmena (směrem zleva)
- Je-li patern např.: „ABRAKADABRA„
- A dostane index 10, B dostane index 8, K = 4, D = 6 a R = 9. Pro ostatní písmena

přiřadíme -1.

- Naimplementujeme tedy funkci Last(char input), která podle písmene vrátí tento index, a pak pro případy 3 a 4 porovnáváme Last(*T(i)*) a *j*, a tím pak víme, jestli i posunout na Last(*T(i)*) (pokud je Last(*T(i)*) < *j*) nebo pouze *i++*

Knuth-Morris-Pratt (KMP) algoritmus

Prohledává text zleva doprava, na rozdíl od brute-force nedělá všechna porovnávání. Pokud narazí na neshodu, posune se o více než jedno písmeno. Najdeme-li kus P (od začátku, tedy prefix), znaky tohoto prefixu odpovídají textu, není třeba je kontrolovat znovu. Konec nalezeného podřetězce může být také obsažen v začátku tohoto podřetězce. Takovou shodou je samozřejmě celá nalezená část P, proto hledáme od P+1. Takže jdeme od konce nalezeného kusu P zleva a zprava, a ve chvíli, kdy nenalezneme shodu, víme, o kolik se můžeme posunout. Toto lze předpočítat do tabulky – pak je vše O(1).

Trie

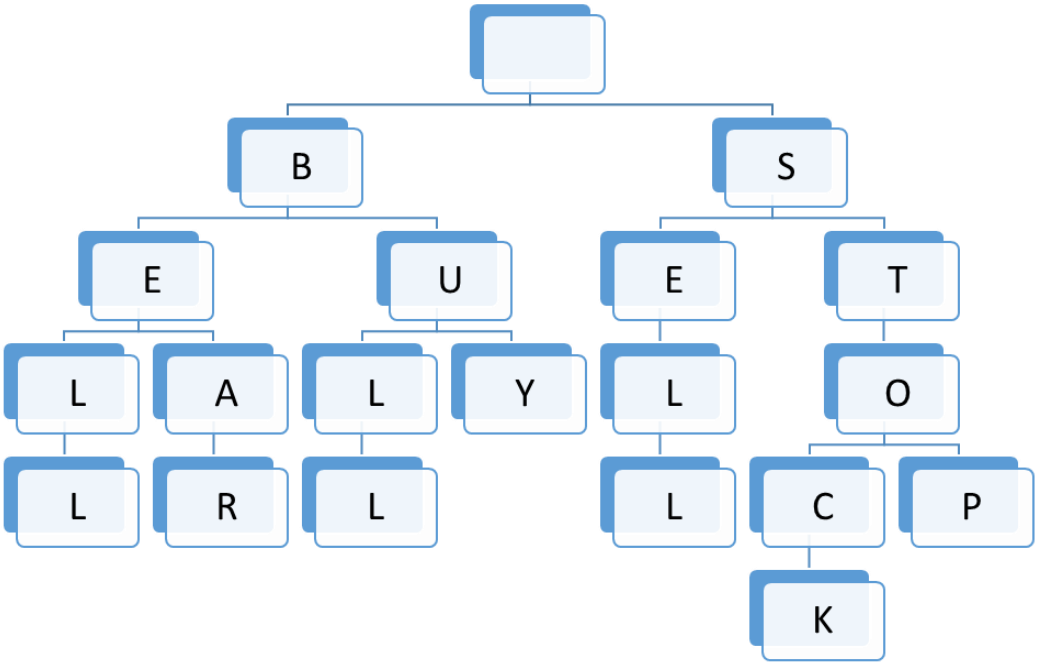
Trie je stromová struktura pro předzpracování textu. V každém uzlu je jedno písmeno. Délka cesty z uzlu k vrcholu určuje pořadí písmene ve slovu.

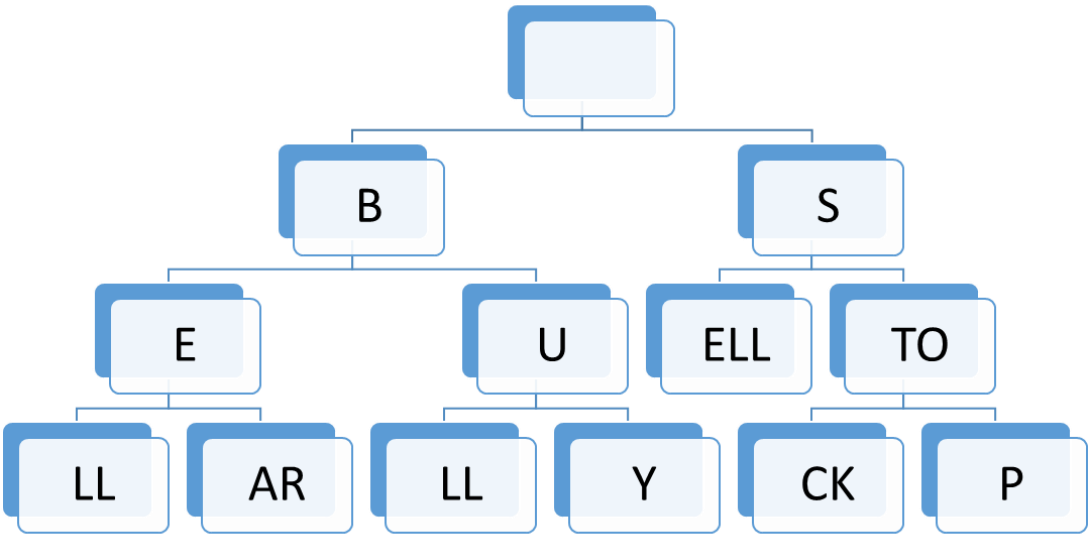
Vyhledávání má časovou náročnost O(dm), kde d je velikost abecedy, m délka slova.

Do struktury Trie je možné ukládat i celý text, pak je v každém uzlu jedno slovo.

Pro úsporu lze definovat tzv. komprimovaný Trie. Strom pak obsahuje uzly alespoň stupně dva (dvě písmena v jednom uzlu).

Příklad: S={BELL, BEAR, BULL, BUY, SELL, STOCK, STOP}





11. Teorie grafů

Graf tvoří uspořádaná dvojice *(V, E)*, kde *V* je množina vrcholů a *E* je množina hran. Každá hrana je určena právě dvěma vrcholy, volitelně pak směrem nebo váhou.

11.1. Typy hran

Existuje několik typů hran:

- Orientovaná – uspořádaná dvojice vrcholů *(u,v)*, kde *u* je počátek, *v* je cíl
- Neorientovaná - uspořádaná dvojice vrcholů *(u,v)*
- Smyčky – hrana začíná a končí ve stejném vrcholu
- Multihrana (násobná, paralelní, rovnoběžná) – mezi vrcholy *(u,v)* vede více hran

11.2. Typy grafů

Stejně jako máme několik typů hran, existuje i několik typů samotných grafů.

- Orientovaný – všechny hrany jsou orientované
- Neorientovaný – všechny hrany jsou neorientované
- Multigraf – obsahuje multihrany

11.3. Terminologie:

- End vertices (or endpoints) of an edge
- Edges incident on a vertex
- Adjacent vertices
- Degree of a vertex
- Parallel edges
- Self-loop
- Path
    - sequence of alternating vertices and edges
    - begins with a vertex
    - ends with a vertex
    - each edge is preceded and followed by its endpoints
- Simple path
- path such that all its vertices and edges are distinct
- Cycle
    - circular sequence of alternating vertices and edges
    - each edge is preceded and followed by its endpoints
- Simple cycle
    - cycle such that all its vertices and edges are distinct

11.4. ADT Graf podporuje operace:

Přístupové operace

- aVertex()
- incidentEdges(v)
- endVertices(e)
- isDirected(e)
- origin(e)
- destination(e)
- opposite(v,e)
- areAdjecent(v,w)

Aktualizační operace

- insertVertex(o)
- insertEdge(v, w, o)
- insertDirectedEdge(v, w, o)
- removeVertex(v)
- removeEdge(e)

Obecné operace

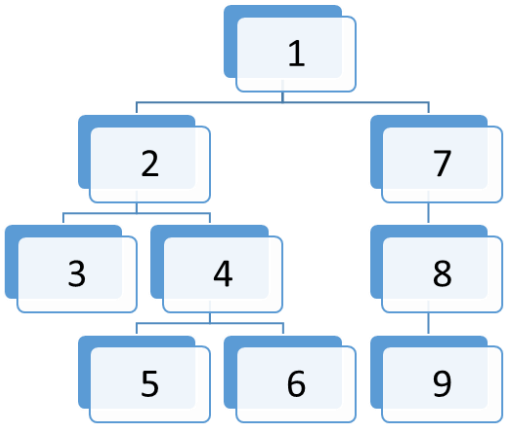
- numVertices()
- numEdges()
- vertices()
- edges()

Strukturu graf je potřeba nějakým způsobem prohledávat. Na výběr máme ze dvou možností – **DFS** (Depth-First Search – prohledávání do hloubky) a **BFS** (Breadth-First Search – prohledávání do šířky)

11.5. Depth-First Search

Prohledávání hloubky je úplný algoritmus (projde každý uzel). Jeho princip spočívá v tom, že expanduje prvního následníka každého vrcholu, pokud jej ještě nenavštívil. Pokud narazí na vrchol, z nějž už nelze dále pokračovat (nemá žádné následníky nebo byli všichni navštíveni), vrací se zpět backtrackingem.

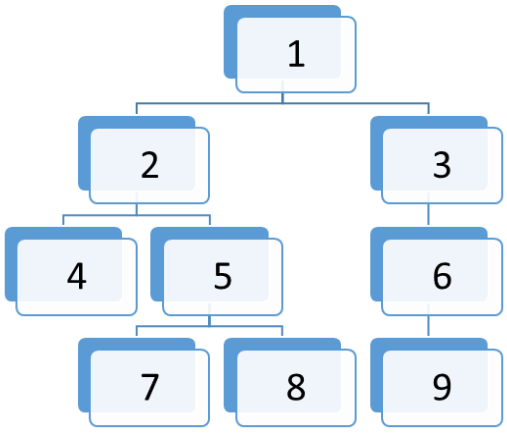
Prochází uzly v pořadí:



11.6. Breadth-First Search

Prohledávání do šířky je podobný algoritmus jako DFS, projde všechny sousedy startovního vrcholu, poté sousedy sousedů atd. až projde celou komponentu souvislosti

Prochází uzly v pořadí:



Základním otázkou z teorie grafů je nalezení nejkratší cesty mezi dvěma vrcholy. K tomu existuje několik algoritmů.

Dijkstrův algoritmus

- Konečný, funguje pouze na kladně ohodnoceném grafu
- $O(|V|^2+|E|)$ – V je počet vrcholů, E počet hran

Bellmanův-Fordův algoritmus

- Graf může obsahovat i záporné hrany
- $O(V \cdot E)$ – pomalejší než Dijsktrův alg.

Floydův-Warshallův algoritmus

- Orientovaný graf s kladnými hranami
- Nalezne nejkratší cestu mezi všemi vrcholy
- Časová náročnost – $O(V^3)$, paměťová náročnost $O(V^2)$

Johnsonův algoritmus

- Orientovaný graf, může mít i záporné hrany
- V řídkých grafech je rychlejší, než Floydův-Warshallův algoritmus
- Dokáže rozpoznat záporný cyklus v grafu a výpočet ukončit
- využívá Bellmanův-Fordův algoritmus, s jehož pomocí přehodnotí hrany tak, aby žádná neobsahovala zápornou hodnotu
- Po přehodnocení hran používá Dijkstrův algoritmus k nalezení nejkratších cest mezi všemi uzly.
- $O(V^2 \log_2(V)+ VE)$

12. Genetické algoritmy

12.1. Výklad

Genetické algoritmy patří mezi evoluční algoritmy a řadí se do oblasti umělé inteligence. Jsou třídou heuristických algoritmů. Využívají znalostí z evoluční biologie k řešení složitých problémů, pro které neexistuje exaktní algoritmus. Napodobuje techniky evoluční biologie:

- Dědičnost
- Mutace
- Přirozený výběr
- Křížení

Princip genetických algoritmů pracuje podle schématu:

- **Inicializace**: Vytvoř nultou populaci (obvykle složenou z náhodně vygenerovaných jedinců)
- **Začátek cyklu:** Vyber (zpravidla zčásti náhodné) z populace několik jedinců s vysokou zdatností
- Z vybraných jedinců vygeneruj novou generaci použitím následujících metod (operátorů):
    - **Křížení:** „prohoď" části několika jedinců mezi sebou
    - **Mutace:**- náhodně změň část jedince
    - **Reprodukce:** kopíruj jedince beze změny
- Vypočti zdatnost těchto nových jedinců
- **Konec cyklu**: Pokud není splněna zastavovací podmínka, tak pokračuj od bodu 2
- **Konec algoritmu**: Jedinec s nejvyšší zdatností je hlavním výstupem algoritmu a reprezentuje nejlepší nalezené řešení.

12.2. Terminologie

- **Fenotyp** – označení jedince
- **Genotyp, genom, chromozom** – reprezentace fenotypu
- **Chromozom** – dělí se na jednotlivé lineárně uspořádané geny (*i*-tý gen chromozomů stejného typu reprezentuje stejnou charakteristiku)
- **Alely** – různé hodnoty genu
- **Fitness hodnota** – z rozmezí 0-1, vyjadřuje kvalitu každého jedince

Každý jedinec může být zakódován (geneticky popsán) různým způsobem. Způsob popsání může být důležitý pro úspěch, či neúspěch řešení konkrétní úlohy.

12.3. Příklad:

Nultá generace (fitness hodnota = počet „1“):

- 0100011011 *f*=0,5
- 0101000100 *f*=0,3
- 1010110000 *f*=0,4
- 1110111000 *f*=0,6

Selekce

- *Vážená ruleta*: $p_i = \frac{f_i}{\sum_1^N f_i}$
  - Pravděpodobnost s jakou bude daný jedinec rodičem

- *Turnajová metoda*
  - Náhodný výběr skupin, z každé skupiny se rodičem stane jedinec s nejvyšší hodnotou fitness.

- *Ořezávání*
  - Seřadíme všechny jedince podle *f* hodnoty, ořízneme část s nízkou hodnotou, ze zbytku vybereme rodiče

- *Náhodný výběr*
  - Nejjednodušší metoda, *f* hodnota nehraje roli při výběru jedince pro rodičovství

- *Křížení*
  - Rodiče si vymění část genetického kódu
  - Nejjednodušší metoda – jednobodové křížení
  - Náhodně vybere místo pro řez
    - X: 010001|1011
    - Y: 111011|1000
  - P: 0100011000 *f*=0,3
  - Q: 111011 1011 *f=0,8*
  - Vícebodové křížení, možnost více než dvou rodičů

- *Mutace*
  - Náhodná změna náhodného genu v jedinci
  - Velmi malá pravděpodobnost

1. 0100011011 ⇒ 0101011011

2. 0101000100 ⇒ 0101100100
3. 1010110000 ⇒ 1010110100
4. 1110111000 ⇒ 1010111000

   - Lze dosáhnout vlastností, které se v původní generaci nevyskytují

Ukončení

- Dosažení maximálního počtu generací (časové omezení)
- Dosažení minimálního potřebného fitness skóre
- Alespoň jeden jedinec dosáhl dostatečně uspokojivého výsledku
- Dosažený přidělený rozpočet (počítačový čas/peníze)
- Po sobě jdoucí iterace nedosahují žádného zlepšení
- Ruční kontrola
- Kombinace výše uvedených kritérií

DATABÁZE

1. Základní pojmy databáze

Databáze (neboli datová základna) je určitá uspořádaná množina informací (dat), většinou tabulka se záznamy, uložená na paměťovém médiu. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim.

Databáze je množina záznamů, kterou shromažďujeme za nějakým konkrétním účelem. Databáze používáme zejména pro ukládání obsáhlých informací. Databázové systémy jsou k dispozici jako součást kancelářských balíků (např. MS Access, OpenOffice.org Base). Tyto systémy jsou k dispozici i jako samostatné programy, které se používají pro tvorbu databází velkého rozhlasu. Například se jedná o MySQL, Oracle a další.

Databáze je soubor dat (informací o objektech reálného světa), která spolu nějakým způsobem souvisí. Data je výraz pro údaje, používané pro popis nějakého jevu nebo vlastnosti pozorovaného objektu. Představují formu prezentace reálných objektů (znaky, symboly, obrázky, fakta, události), odrážejí tedy stav reality v určitém časovém okamžiku. Informace je zpráva, že nastal určitý jev. Vzniká přiřazením významu datům a existuje ve vztahu k příjemci. Slouží k informování o změnách ve vnímané realitě. S databázemi se v běžném životě setkáváme velmi často. Uvádíme běžné použití databází velkého rozsahu:

- databáze jízdních řádů,
- databáze státní správy,
- informační systémy bank, škol, úřadů
- nemocniční systémy evidence pacientů,
- databáze knihoven.

1.1. Systém řízení báze dat

Systém řízení báze dat (zkracováno na SŘBD či DBMS podle anglického database management system) je softwarové vybavení, které zajišťuje práci s databází, tzn. tvoří rozhraní mezi aplikačními programy a uloženými daty. Někdy se tento pojem zaměňuje s pojmem databázový systém. Databázový systém však je SŘBD dohromady s bází dat.

Entita

Libovolný objekt (osoba, zvíře, věc či jev) reálného světa, který je zachycen v datovém modelu. Entita musí být rozlišitelná od ostatních entit a existovat nezávisle na nich.

Data

Výraz pro údaje používané pro popis nějakého jevu nebo vlastnosti pozorovaného objektu. Data se získávají měřením nebo pozorováním, a lze je dělit na data spojitá a data atributivní. Data spojitá se přitom vztahují k nějaké spojité stupnici, zatímco data atributivní nikoliv

Informace

Informace jsou data, která nám přinášejí nové poznatky.

Záznamy a atributy

Řádky tabulky s hodnotami atributů pro jeden objekt (entitu), které se musí od sebe lišit. Atributy jsou sloupce tabulky, Atributy mají určen svůj konkrétní datový typ a doménu, což je množina přípustných hodnot daného atributu. Řádek je řezem přes sloupce tabulky a slouží k vlastnímu uložení dat.

Atributy, pole

Vlastnosti, které se u objektů (entit) sledují:

- tvoří sloupce tabulky
- mohou nabývat různých hodnot
- pole jsou určitého datového typu (číslo, text, datum, ...)

Primární klíč

Jednoznačně identifikuje záznam (řádek tabulky).
Je to takový atribut (pole), který má pro každou entitu jedinečnou hodnotu, např. rodné číslo, většinou je to pomocné pole s identifikačním číslem záznamu (ID).

Cizí klíč

Takový atribut, který je v jiné tabulce primárním klíčem.

Index

Jedná se o způsob řazení tabulky.

- pořadí záznamů se v tabulce během „života“ databáze nemění; index pomáhá k rychlému hledání dat v tabulce
- index vytvoří pomocný soubor s řazením tabulky podle určitého pole
- k jedné tabulce může být více indexů s řazením podle různých atributů
- primární klíč je vždy indexem

1.2. Struktura databáze

Nejpoužívanějšími databázemi jsou relační databáze. V nich jsou data ukládána v menších tabulkách, aby se zajistila minimální redundance (nadbytečnost) dat. Tabulky jsou vzájemně propojeny pomocí relací. Relace určují vztahy mezi tabulkami a zjišťují provázanost jednotlivých tabulek. Každá z těchto tabulek by měla obsahovat data týkající se pouze jednoho druhu objektu (např. tabulka objednávek, klientů, cen, zboží atd.). Pro vytvoření dobré databáze je nutné nejdříve navrhnout správnou strukturu jednotlivých tabulek. Tyto tabulky je pak nutné propojit pomocí relací. Tabulky tvoří základ celé struktury databáze. Základní pravidla pro návrh tabulek jsou následující:

- každá informace by měla být v databázi obsažena pouze jednou,
- každá tabulka by měla obsahovat informace o jednom typu objektu,
- při návrhu tabulek by se měl vzít do úvahy rozsah budoucích dat.

1.3. Databázová tabulka

V jedné tabulce by měly být informace o jednom typu objektu. Databázová tabulka je podobná běžné tabulce. Řádky obsahují záznamy (o jednom objektu) a sloupce označujeme položky či pole. Polem je totiž někdy zván i průsečík určitého řádku a sloupce, který obsahuje jedinou hodnotu (datový prvek).

Zaměstnanci

ID_zamestn	Jméno	Příjmení	Datum naro	Pohlaví	Telefonní čí	ID_funkce
1	Tomáš	Novák	28.4.1980	muž	723 123 456	F_03
2	Josef	Koblížek	15.3.1976	muž	728 452 123	F_01
3	Petra	Maková	5.2.1985	žena	724 556 115	F_02
4	Václav	Sýkorka	30.6.1970	muž		F_06
5	Denisa	Rosolová	12.12.1956	žena		F_05
6	Michal	Aspik	3.6.1976	muž		F_04
7	Dominik	Kokeš	14.2.1981	muž		F_04
8	Tereza	Železná	25.10.1978	žena		F_02
9	Vladimíra	Mimořádná	17.11.1989	žena		F_02
10	Jakub	Pekelník	5.12.1973	muž		F_05

V uvedené tabulce zaměstnanci tvoří řádky záznamy s informacemi o jednotlivých zaměstnancích. Sloupce představují pole, ve kterých vidíme vždy jeden typ dat (text, datum číslo). Každý sloupec (položka) má název, zvolený datový typ (např. text, číslo, ano/ne, datum a čas) a velikost. Lze přiřadit i další vlastnosti (formát, výchozí hodnotu, atd.). Více k této problematice ve videonávodech.

1.4. Primární klíč

Ve většině případů potřebujeme každý vložený záznam do tabulky jednoznačně identifikovat. K tomu slouží tzv. primární klíč. Primární klíč je takové pole, které je určeno pro zajištění jednoznačné identifikace jednotlivých záznamů v tabulce. Primární klíč je obvykle tvořen jedním pole (tzv. jednoduchý primární klíč), ale může být tvořen i více poli tabulky (tzv. složený primární klíč).

V každé tabulce může být pouze jeden primární klíč. S pomocí primárního klíče se rychleji vyhledávají informace a také se s ním vytváří relace s ostatními tabulkami. Hodnoty v poli primárního klíče musí být jedinečné pro každý uvedený záznam. Primární klíč je jedním z indexů. Pro rychlejší vyhledávání a třídění záznamů podle určitého pole tabulky je vhodné používat indexování polí (indexy). Pokud potřebujeme vyhledávat podle jiného sloupce tabulky než je primární klíč, tak mu nastavíme index. S pomocí nastaveného indexu se zrychlí řazení, vyhledávání a editace hodnot v tabulkách.

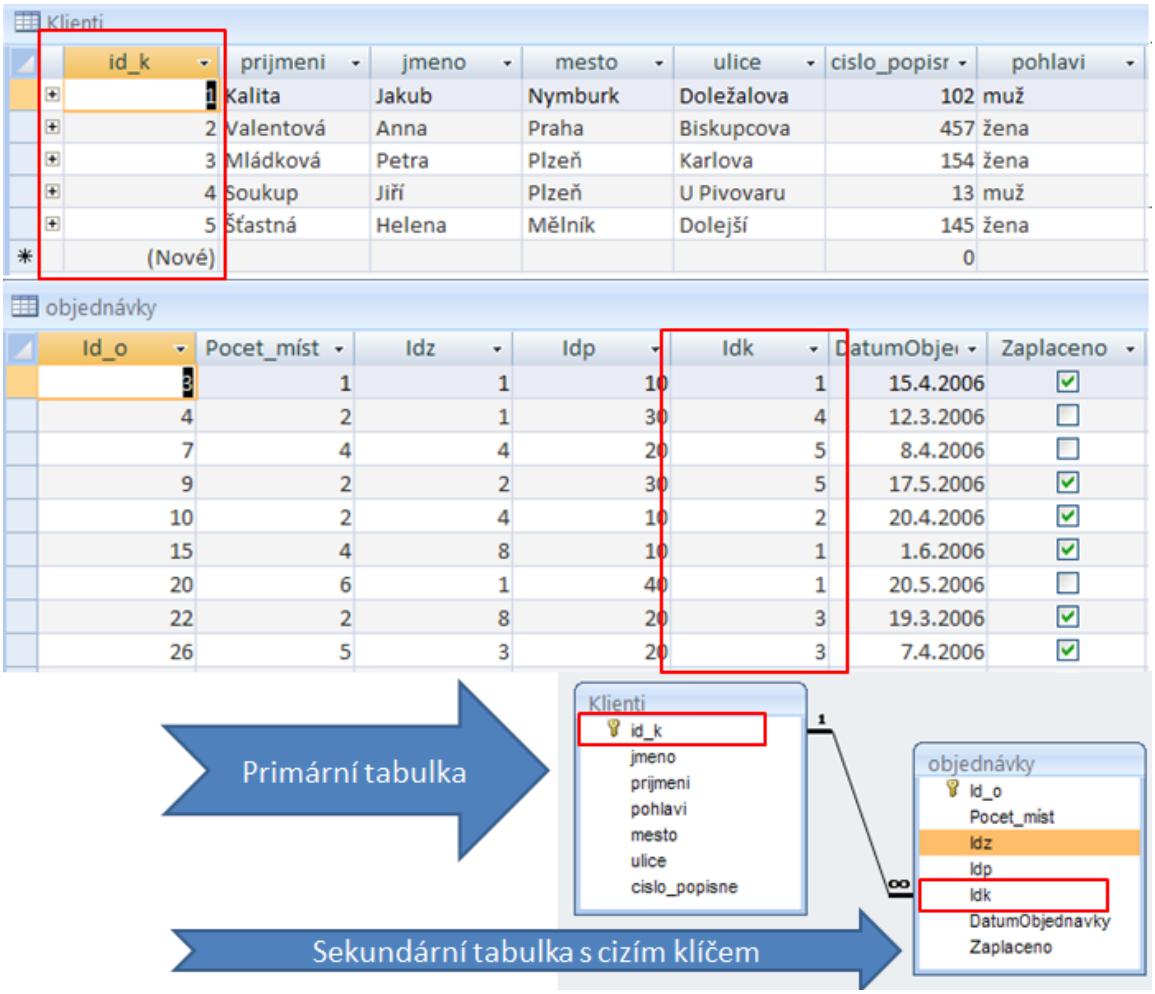
1.5. Relace

V relační databázi jsou jednotlivé tabulky propojeny pomocí relací. Hlavním účelem relací mezi tabulkami je omezení výskytu redundantních (nadbytečných) dat. Data by se neměla v různých tabulkách v rámci jedné databáze opakovat. Relace je postavena na vazbě stejných hodnot mezi unikátním polem (primárním klíčem) jedné tabulky a odpovídajícím polem jiné tabulky. Pro vytvoření relace mezi dvěma tabulkami je nutné mít v každé tabulce speciální pole. V jedné z těchto tabulek je to primární klíč. Takovou tabulku označujeme jako primární tabulku. Ve druhé tabulce vytváříme speciální pole pro účely relace. Toto pole označujeme jako cizí klíč, obsahuje hodnoty primárního klíče z jiné tabulky. Tabulka obsahující cizí klíč je označována jako sekundární tabulka.

1.6. Primární a sekundární tabulka

Pole primárního klíče obsahuje pouze unikátní hodnoty. Pole cizího klíče může obsahovat i stejné hodnoty. Pole primárního i cizího klíče musí obsahovat pro správné vytvoření relace stejné hodnoty. Na obrázku na následující straně uvádíme primární tabulku, která obsahuje záznamy o jednotlivých zákaznících. Primárním klíčem je první pole, které obsahuje číslo zákazníka. Tato tabulka je propojena se sekundární tabulkou, která je tvořena záznamy o objednávkách zákazníků. Cizím klíčem v sekundární tabulce je pole obsahující číslo zákazníka, který objednávku provedl. Z uvedeného příkladu vidíme, že jednomu záznamu z primární tabulky odpovídá jeden nebo více záznamu ze sekundární tabulky. Jinými slovy jeden zákazník může provést jednu nebo více ob-

jednávek. Jedná se o typ relace 1:N.



1.7. Typy relací

Rozlišujeme tři základní typy relací:

- **Relace typu 1:1 (výjimečný)** – jednomu záznamu primární tabulky odpovídá právě jeden záznam v sekundární tabulce. Například jedné osobně je přiděleno právě jedno rodné číslo.
- **Relace typu 1:N (nejčastější)** – jednomu záznamu primární tabulky odpovídá jeden nebo více záznamů v sekundární tabulce. Tento příklad jsme popsali ve výše uvedeném příkladu se zákazníky a objednávkami.
- **Relace typu M:N (velmi často)** – jednomu nebo více záznamům primární tabulky odpovídá jeden nebo více záznamů v sekundární tabulce. Tyto relace řešíme pomocí spojovací tabulky, kterou pomocí dvou relací typu 1:N propojíme s původními tabulkami.

1.8. Referenční integrita

Referenční integrita udržuje neporušenost relací mezi tabulkami. Nedovolí nám vložit do sekundární tabulky záznam, který by neměl odpovídající záznam v primární tabulce. Dále si pohlídá si změnu hodnot cizího klíče při změně primárního klíče. V rámci referenční integrity je možné nastavit pravidla pro odstraňování záznamů.

1.9. Obsluha databáze

Profesionální databáze obsahují velké množství důležitých informací. Osoby, které pracují s takovou databází, můžeme rozdělit do několika skupin:

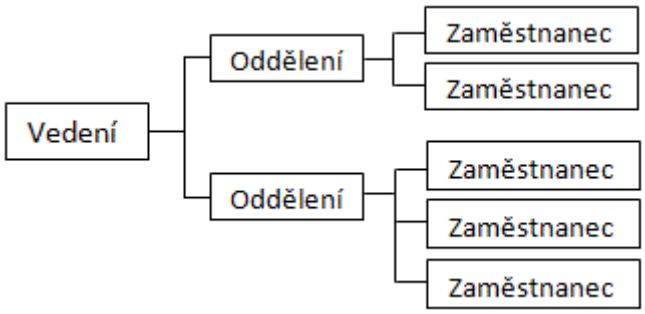
- databázový specialista navrhuje a vytváří profesionální databáze,
- uživatel zadává data, udržuje data a získává informace z databáze,
- správce databáze poskytuje uživatelům oprávnění přístupu k určitým datům, je odpovědný za obnovu databáze po její havárii nebo výskytu závažné chyby.

2. Databázové modely

Z hlediska způsobu ukládání dat a vazeb mezi nimi dělíme databáze do základních typů:

2.1. Hierarchický model dat

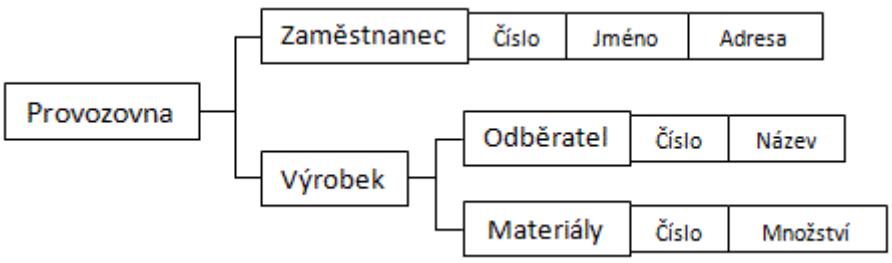
Data jsou organizována do stromové struktury. Každý záznam představuje uzel ve stromové struktuře, vzájemný vztah mezi záznamy je typu rodič/potomek. Nalezení dat v hierarchické databázi vyžaduje navigaci přes záznamy směrem na potomka, zpět na rodiče nebo do strany na dalšího potomka. Největšími nevýhodami hierarchického uspořádání je složitá operace vkládání a rušení záznamů a v některých případech i nepřirozená organizace dat.



Obr. 1 - Hierarchický model

2.2. Síťový model dat

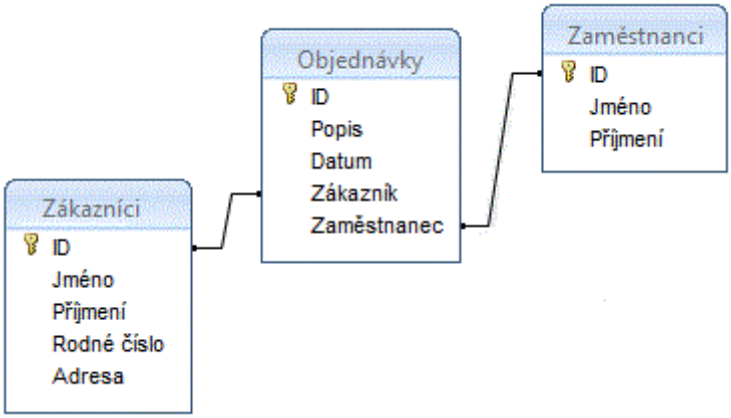
Síťový model dat je v podstatě zobecněním hierarchického modelu, který doplňuje o mnohonásobné vztahy (sety). Tyto sety propojují záznamy různého či stejného typu, přičemž spojení může být realizováno na jeden nebo více záznamů. Přístup k propojeným záznamům je přímý bez dalšího vyhledávání, k dispozici jsou operace: nalezení záznamu podle klíče, posun na prvního potomka v dílčím setu, posun stranou na dalšího potomka v setu, posun nahoru z potomka na jeho rodiče v jiném setu. Nevýhodou síťové databáze je zejména nepružnost a obtížná změna její struktury.



Obr. 2 - Síťový model

2.3. Relační model dat

Relační databázový model je z uvedených nejmladší a zároveň nejpoužívanější. V současnosti je nejčastěji využíván u komerčních SŘBD. Model má jednoduchou strukturu, data jsou organizována v tabulkách, které se skládají z řádků a sloupců. V těchto tabulkách jsou prováděny všechny databázové operace.



Obr. 3 - Relační model

Databáze dle relačního modelu musí splňovat tyto dvě vlastnosti:

- Databáze je chápana uživatelem jako množina relací a nic jiného.
- V relačním SŘBD jsou k dispozici minimálně operace selekce, projekce a spojení, aniž by se vyžadovaly explicitně předdefinované přístupové cesty pro realizaci těchto operací.

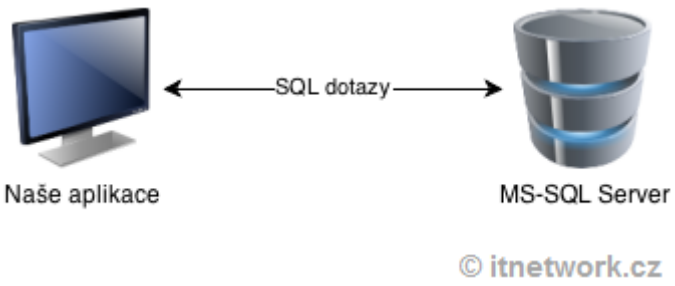
2.4. Objektové databáze

Kromě databází relačních existují i již zmíněné databáze objektové. Ty řeší problém neslučitelnosti objektového a relačního přístupu. Poskytují stejný komfort, jako ORM, ale vnitřně není třeba data převádět do tabulek, ukládají se rovnou jako objekty. Teoreticky neexistuje výkonnostní ani jiný důvod, proč by neměly nahradit databáze relační. V praxi se ale bohužel téměř nepoužívají a můžeme jen doufat, že se to časem změní.

2.5. Připojená aplikace

Přístup připojené aplikace použijeme ve chvíli, kdy potřebujeme data často v reálném čase číst nebo měnit. Pomocí tříd DataReader, Command a Connection posíláme databázi přímo příkazy v jazyce SQL a dostáváme výsledky.

Situaci je znázorněna na obrázku:



2.6. Objektové databáze

Univerzální databázové systémy (database management systems – DBMS) v současné době představují nejrozšířenější prostředek pro uchovávání a manipulaci s daty komerčních aplikací. Zajišťují relativně jednoduchou správu rozsáhlých databází, integritu, efektivní vyhledávání a zpracovávání informací, odolnost proti poruchám a výpadkům, souběžný víceuživatelský přístup a další výhody.

Relační datový model – nejběžnější model používaný v dnešních databázových systémech – omezuje strukturu a vztahy uchovávaných dat na pouhou množinu tabulek nad předdefinovanou množinou základních datových typů. Nespornou výhodou tohoto modelu je jednoduchost a v důsledku toho i snadná standardizace a přenositelnost. Nevýhodou však je rozdílnost schématu reálných dat od vnitřního tabulkového modelu databáze. Veškeré vztahy mezi daty lze reprezentovat pouze tabulkami, což u aplikace s komplikovanějším datovým modelem vede k množství tabulek vzájemně provázaných pomocnými odkazy, tzv. klíči. Tím dochází ke ztrátě přehlednosti, databáze se stává hůře spravovatelnou a budoucí změny v datovém modelu aplikace nutí programátory k citelným zásahům do jejich tabulkové reprezentace. Relační databáze jsou a zřejmě i nadále zůstanou hlavním prostředkem pro správu dat komerčních aplikací, které jsou charakteristické velkým objemem údajů s jednoduchou strukturou. Řada aplikací, například z oblasti designu, multimédií, geografických systémů apod., však potřebuje takový datový model, který umožní lepší korespondenci mezi složitými reálnými daty a jejich reprezentací v databázovém systému. Tímto modelem je objektový model dat. Objektový model dat v databázových systémech vychází ze známých principů objektově orientovaného modelování a programování. Je však dále obohacen o techniky perzistence, reprezentace vztahů, dotazování, transakčního přístupu apod.

2.7. Základy objektové orientace. Objekty a třídy

Objektově orientovaný model je založen na dekompozici informací z reálného světa na tzv. objekty. Objektem se rozumí každá (i strukturovaná) entita, která je jednoznačně a nezávisle identifikovatelná v rámci určitého kontextu okolního světa. Objekt tak má jednoznačnou identitu, každé dva i jinak datově shodné objekty jsou vzájemně odlišitelné. Identita objektu je určena identifikátorem (object identifier – oid), který je generovaný systémem, unikátní, neměnný po dobu existence objektu, skrytý pro programátora i koncového uživatele. Objekty jsou charakterizovány pomocí tříd. Třída je abstraktní popis objektu, určuje datové složky objektu a operace (nazývané metody), které lze nad objektem provádět. Každý objekt je instancí nějaké třídy, od jedné třídy je možné instanciovat obecně neomezený počet strukturálně shodných objektů. Dále rozlišujeme rozhraní třídy.

2.8. Literály

Kromě objektů se v rámci objektově orientovaného modelu zavádí i pojem literálu. Literál je datová entita určitého datového typu, která však na rozdíl od objektu nemá vlastní identitu. Literály se obvykle vyskytují jako datové atributy objektů. Množina operací nad datovým typem literálu je pevně stanovená, není možné ji měnit. S objekty a literály souvisí pojem proměnlivosti (mutability). Proměnlivost je chápána jako schopnost měnit data při zachování identity – v tomto smyslu jsou objekty proměnlivé, protože je možné měnit hodnoty jejich datových složek a přitom si ponechávají původní identitu. Naproti tomu literály proměnlivé nejsou.

2.9. Operace

Existuje několik základních typů operací nad objekty. Každý objekt má jeden či více konstruktorů. Účelem konstruktoru je inicializovat objekt v okamžiku jeho vytvoření. Dále je součástí každého objektu tzv. destruktor. Destruktor je volán v okamžiku rušení objektu a jeho cílem je provést úklid objektu před jeho odstraněním. Důležitou operací nad objekty je kopírování. Rozlišují se tzv. mělké (shallow) a hluboké (deep) kopie. V případě mělké kopie dojde ke zkopírování atributů objektu, avšak všechny případné odkazy na jiné objekty dále ukazují na stejné objekty jako v originálním objektu. Naproti tomu u hluboké kopie dojde nejen ke zkopírování atributů kopírovaného objektu, ale současně se vytvoří i kopie objektů, na které se původní objekt odkazoval. Dalšími typickými operacemi jsou metody pro zjišťování a přiřazování hodnot atributů, metody provádějící výpočty a manipulace s atributy objektu, metody produkující uživatelský výstup atd.

Objektový datový model jsou data uložené v objektové struktuře. Jde většinou o objektovou mezivrstvu mezi kódem a databází, do které se nasypou data, s kterými pak aplikace pracuje a nezatěžuje dotazy DB server.

Objektově relační datový model

Objektově-relační datový model je klasická tabulková databáze rozšířená o **abstraktní datové typy** (ADT). ADT jsou uživatelem definované typy skládající se ze základních datových typů databáze. Čímž porušuje 1NF??

- Objektové rysy jsou dnes implementovány ve všech velkých SŘBD.
- Objektové typy a jejich metody jsou uloženy spolu s daty v databázi, programátor tedy nemusí vytvářet podobné struktury v každé aplikaci.
- Programátor může přistupovat k množině objektů, jako by se jednalo o jeden objekt.
- Objekty mohou jednoduše reprezentovat vazby, kdy jedna entita se skládá z jiných entit (bez nutnosti použít vazeb).
- Metody jsou spouštěny na serveru – nedochází k neefektivnímu přenosu dat po síti.

Objektové datové typy

mohou obsahovat:

- data – atributy
- operace – metody
- Specifické pro ORŠRBD je to, že se na data můžeme dívat jak z relačního (záznamy, operace) tak objektového pohledu (objekty, metody).

Typy metod

- **Členské metody** – jsou volány nad konkrétním objektem.
- **Statické metody** – jsou volány nad datovým typem.
- **Konstruktor** – pro každý objektový typ je definován implicitní konstruktor.

3. Integrita databáze

Integrita databáze znamená, že data v ní uložená jsou konzistentní vůči definovaným pravidlům. Lze zadávat pouze data, která vyhovují předem definovaným kritériím (např. musí respektovat datový typ nastavený pro daný sloupec tabulky, či další omezení hodnot přípustných pro daný sloupec). K zajištění integrity slouží integritní omezení. Jedná se o nástroje, které zabrání vložení nesprávných dat či ztrátě nebo poškození stávajících záznamů v průběhu práce s databází. Například je možné zajistit mazání dat, která již ztratila svůj význam - například smažeme-li uživatele, odstraní se i zbytek jeho záznamů v ostatních databázových tabulkách.

3.1. Druhy integritních omezení

- **Entitní integritní omezení** – povinné integritní omezení, které zajišťuje úplnost primárního klíče tabulky (zamezí uložení dat, jež by v těchto polích byla stejná jako v nějakém jiném řádku tabulky)
- **Doménová integritní omezení** – zajišťují dodržování datových typů/domén definovaných u sloupců databázové tabulky
- **Referenční integritní omezení** – zabývají se vztahy dvou tabulek, kde jejich relace je určena vazbou primárního a cizího klíče
- **Aktivní referenční integrita** – definuje činnosti, které databázový systém provede, pokud jsou porušena některá pravidla

3.2. Dodržování integritních omezení

V zásadě existují tři způsoby, jak zajistit dodržování integritních omezení:

- Umístění jednoduchých mechanismů pro dodržování integritních omezení **na straně databázového serveru**
    - jedná se o nejlepší způsob z hlediska ochrany dat
    - uživateli však obvykle přináší delší odezvu systému a nelze vždy zajistit jejich přenositelnost na jiný databázový systém
- Umístění ochranných mechanismů **na straně klienta**
    - pro komfort a nezávislost na databázovém systému je nejlepší volbou nutnost kontrolních mechanismů pro každou operaci může způsobit chyby u aplikací a v případě většího počtu aplikací je potřeba je opravit na více místech

- Samostatné programové moduly **na straně serveru**
    - v moderních databázových systémech jsou pro tento účel implementovány tzv. triggery = samostatné procedury, které lze spouštět automatizovaně před a po operacích manipulujících s daty
    - tento způsob umožňuje implementaci i složitých integritních omezení
    - nevýhody opět přináší provádění na serveru i velmi omezená možnost přenesení na jiný databázový systém

Ideálním řešením je kombinace předchozích v závislosti na konkrétních podmínkách. Kontroly integritních omezení se zpravidla provádějí po každé provedené operaci, což snižuje nároky na server. Není nutno nijak zaznamenávat, které kontroly mají být provedeny později. Složitější integritní omezení však vždy nelze takto ověřit, proto je možné kontrolovat dodržení pravidel až po dokončení celé transakce.

3.3. Vztahy mezi tabulkami

Relace slouží ke svázání dat, která spolu souvisejí a jsou umístěny v různých databázových tabulkách.

V zásadě rozlišujeme čtyři typy vztahů:

- mezi daty v tabulkách není **žádná spojitost**, proto nedefinujeme žádný vztah
- **1:1** (záznamu odpovídá právě jeden záznam v jiné databázové tabulce a naopak)
- **1:N** (přiřazuje jednomu záznamu více záznamů z jiné tabulky)
    - jedná se o nejpoužívanější typ relace, jelikož odpovídá mnoha situacím v reálném životě
- **M:N** (umožňuje několika záznamům z jedné tabulky přiřadit několik záznamů z tabulky druhé)
    - tento vztah bývá z praktických důvodů nejčastěji realizován kombinací dvou vztahů 1:N a 1:M, které ukazují do pomocné, tzv. vazební tabulky složené z kombinace obou použitých klíčů

3.4. Normální formy

Pod pojmem normalizace rozumíme proces zjednodušování a optimalizace navržených struktur databázových tabulek. Hlavním cílem je navrhnout databázové tabulky tak, aby obsahovaly minimální počet redundantních dat. Správnost navržení struktur lze ohodnotit některou z následujících normálních forem.

Nultá normální forma (0NF)

- tabulka obsahuje alespoň jeden sloupec (atribut), který může obsahovat více druhů hodnot

První normální forma (1NF)

- všechny sloupce tabulky nelze dále dělit na části nesoucí nějakou informaci -> prvky musí být atomické
- jeden sloupec neobsahuje složené hodnoty.

Druhá normální forma (2NF)

- tabulka obsahuje pouze sloupce, které jsou závislé na celém klíči

Třetí normální forma (3NF)

- tabulka je ve třetí normální formě, pokud neexistují žádné závislosti mezi neklíčovými sloupci

Čtvrtá normální forma (4NF)

- sloupce v tabulce popisují pouze jeden fakt nebo jednu souvislost

Pátá normální forma (5NF)

- přidáním libovolného nového sloupce by se tabulka rozpadla na více tabulek

3.5. Databázová integrita

Integrita databáze znamená, že databáze vyhovuje zadaným pravidlům – integritním omezením. Tato integritní omezení jsou součástí definice databáze, a za jejich splnění zodpovídá systém řízení báze dat.

Integritní omezení se mohou týkat jednotlivých hodnot vkládaných do polí databáze (například známka z předmětu musí být v rozsahu 1 až 5), či může jít o podmínku na kombinaci hodnot v některých polích jednoho záznamu (například datum narození nesmí být pozdější než datum úmrtí). Integritní omezení se může týkat i celé množiny záznamů daného typu – může jít o požadavek na unikátnost hodnot daného pole či kombinace polí v rámci celé množiny záznamů daného typu, které se v databázi vyskytují (například číslo průkazu v záznamech o osobách).

Velmi často používaným integritním omezením v relačních databázích je tzv. referenční integrita. Jedná se o požadavek, aby pro pole záznamu, jež má obsahovat odkaz na jiný záznam někde v databázi, takový odkazovaný záznam skutečně existoval, tedy aby takový odkaz nevedl „do prázdna“ a nejednalo se o tzv. databázového sirotka.

Integritní omezení

Je třeba zajistit, aby se do databáze dostaly jen takové záznamy, které odpovídají vztahům v reálném světě (např. atribut věk by neměl nabývat záporných hodnot). K ošetření takových případů se používají integritní omezení, která slouží ke stanovení určitého rozmezí hodnot, jakých může konkrétní atribut nabývat. Integritní omezení specifikují, jaká omezení a vnitřní vztahy musí splňovat data v databázi.

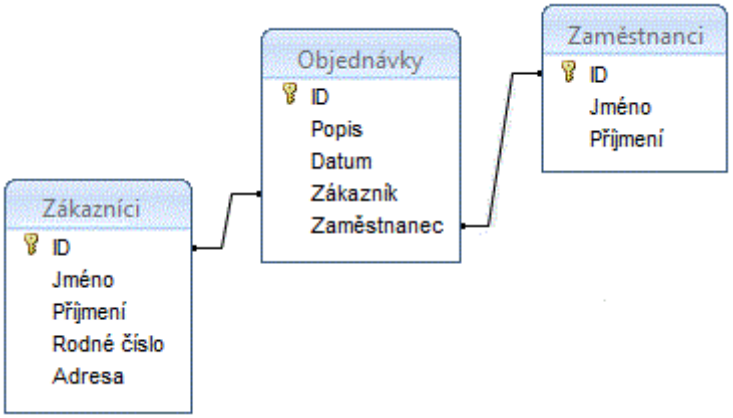
Relace, které vyhovují integritním omezením, jsou přípustné.

Relační databázový systém

- musí plnit tyto tři základní funkce:
  - umožnit definovat typ dat
  - umožnit pracovat s daty
  - musí umožňovat efektivní správu a analýzu uložených dat

- systém by měl obsahovat prostředky umožňující provést požadované činnosti s daty - např. řazení dat, filtrování, výpočty s daty, řídit správu dat
- systém kontroluje zejména přístup k datům, tedy kdo je oprávněn k datům přistupovat a kdo může provádět jejich aktualizaci
- systém řídí sdílení dat mezi více uživateli

4. Relační databázový model



Obr. 3 - Relační model

Databáze dle relačního modelu musí splňovat tyto dvě vlastnosti:

- Databáze je chápana uživatelem jako množina relací a nic jiného.
- V relačním SŘBD jsou k dispozici minimálně operace selekce, projekce a spojení, aniž by se vyžadovaly explicitně předdefinované přístupové cesty pro realizaci těchto operací.

4.1. 12 pravidel pro relační SŘBD

1. Informační pravidlo:
Všechny informace v relační databázi jsou vyjádřeny explicitně na logické úrovni jediným způsobem - hodnotami v tabulkách.

2. Pravidlo jistoty:
Všechna data v relační databázi jsou zaručeně přístupná kombinací jména tabulky s hodnotami primárního klíče a jménem sloupce.

3. Systematické zpracování nulových hodnot:
Nulové hodnoty jsou plně podporovány relačním SŘBD pro reprezentaci informace, která není definována a to nezávisle na datovém typu.

4. Dynamický on-line katalog založený na relačním modelu:
Popis databáze je vyjádřen na logické úrovni stejným způsobem jako zákaznická data, takže autorizovaný uživatel může aplikovat stejný relační jazyk ke svému dotazu jako uživatel při práci s daty.

5. **Obsáhlý datový podjazyk:**
Relační systém může podporovat několik jazyků a různých módů použitých při provozu terminálu. Nicméně musí být nejméně jeden příkazový jazyk s dobře definovanou syntaxí, který obsáhle podporuje definici dat, definici pohledů, manipulaci s daty jak interaktivně, tak programem, integritní omezení, autorizovaný přístup k databázi, transakční příkazy apod.

6. **Pravidlo vytvoření pohledů:**
Všechny pohledy, které jsou teoreticky možné, jsou také systémem vytvořitelné.

7. **Schopnost vkládání, vytvoření a mazání:**
Schopnost zachování relačních pravidel u základních i odvozených relací je zachována nejen při pohledu na data, ale i při operacích průniku, přidání a mazání dat.

8. **Fyzická datová nezávislost:**
Aplikační programy jsou nezávislé na fyzické datové struktuře.

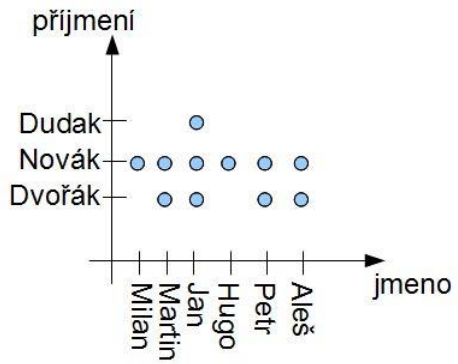
9. **Logická datová nezávislost:**
Aplikační programy jsou nezávislé na změnách v logické struktuře databázového souboru.

10. **Integritní nezávislost:**
Integritní omezení se musí dát definovat prostředky relační databáze nebo jejím jazykem a musí být schopna uložení v katalogu a nikoliv v aplikačním programu.

11. **Nezávislost distribuce:**
Relační SŘBD musí být schopny implementace na jiných počítačových architekturách.

12. **Pravidlo přístupu do databáze:**
Jestliže má relační systém jazyk nízké úrovně, pak tato úroveň nemůže být použita k vytváření integritních omezení a je nutno vyjádřit se v relačním jazyce vyšší úrovně.

4.2. Relační datový model



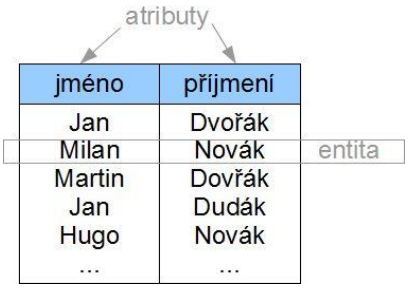
Tabulka s dvěma atributy znázorněná jako relace

Relační datový model je způsob uchování dat v tabulkách. Relační se mu říká proto, že se tabulka se definuje přes relaci.

Relace je tedy v podstatě tabulka. Definována je jako podmnožina kartézského součinu domén.

Relace na obrázku vpravo je tedy podmnožina součinu množin {Dudák, Novák, Dvořák} x {Milan, Martin, Jan, ..., Aleš}

Na rozdíl od **matematické relace** se ta databázová mění v čase (přidáváním a odebíráním prvků relace). Kromě základních množinových operací se u databázové relace setkáme s operaci **selekce** - výběr řádků a **projekce** -výběr sloupců.



Relace z předchozího obrázku zobrazena tabulkou

Doména je množina všech hodnot, kterých může nabývat atribut. Jinak řečeno obor hodnot atributu. V praxi je doména dána integritním omezením (IO).

*Doména atributu Přijmení z obrázků je množina {Dudak, Novák, Dvořák}. *pozn.*

Atribut je vlastnost entity. Z pohledu tabulky jde o sloupec.

Relační schéma můžeme chápat jako strukturu tabulky (atributy a domény).

Příklad pro tabulku (relaci) Učitel:

Atributy: ID, jméno, příjmení, funkce, kancelář
Domény:
D1 - tři písmena z příjmení, tří cifry pořad. čísla
D2 - kalendář jmen
D3 - množina příjmení
D4 - množina funkcí (asistent, vědec, učitel,...)

D5 - A101, A102, … A160

Relační schéma: Učitel (ID, jméno, příjmení, funkce, kancelář)

Relace: Učitel = {(nov001,lukas,novak,vědec,A135),(kom123,jan,komensky,učitel,A111),…}

Definováno je jako R(A,f), kde A je množina atributů (A1,A2,…An) a funkce f(Ai)=Di přiřazuje atributu doménu.

4.3. Vlastnosti relačního datového modelu

Z definice relace vyplývají tyto jejich tabulkové vlastnosti:

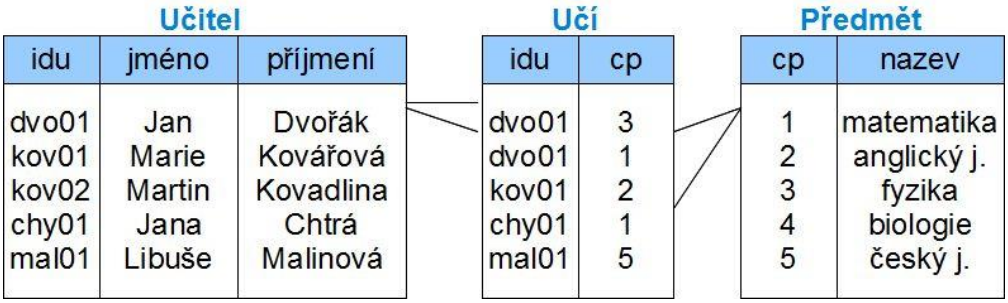
- homogenita sloupců (prvky domény)
- každý údaj (hodnota atributu ve sloupci) je atomickou položkou
- na pořadí řádků a sloupců nezáleží (jsou to množiny prvků/atributů)
- každý řádek tabulky je jednoznačně identifikovatelný hodnotami jednoho nebo několika atributů (primárního klíče)

4.4. Vazby v relačním modelu

Obecně se vazby v relačním modelu realizují pomocí další relace (tabulky). Jedná se o tzv. **vazební tabulku**. Ta **obsahuje** ty atributy relací (tabulek, které se vazby účastní), které jednoznačné identifikují jejich entity - **primární klíče**. Obsahuje-li tabulka atribut, který slouží jako primární klíč v jiné tabulce, pak obsahuje **cizí klíč**. Vazební tabulka tedy obsahuje cizí klíče.

Na obrázku dole máme zobrazenou relaci Učitel s primárním klíčem *idu* a relaci Předmět s primárním klíčem *cp*. K vyjádření vztahu *Učitel UČÍ Předmět* byla vytvořena nová relace Učí, která obsahuje dva cizí klíče (*idu* odkazující na učitele a *cp* odkazující na entitu předmětu). Nyní tedy můžeme přes vazební tabulku pospojovat učitele s předměty, podle toho kdo co učí.

A proč jsme do tabulky Učitel nepřidali jen atribut předmět? Jednoduše proto, že učitel může učit více předmětů. Mezi učitelem a předmětem je vazba M:N. Takže ani kdybychom přidali do tabulky Předmět atribut učitel, bychom tento vztah nevyřešili (předmět může být učen více učiteli). Pro vztah 1:N by to však šlo a v praxi se to tak i dělá. Takže **vazební tabulka je nutná jen k realizací vztahů M:N**.



Ukázka vazební tabulky pro vztah Učí mezi tabulkami Učitel a Předmět. Vztah je M:N, tedy že jeden učitel může učit N předmětů a jeden předmět může být učen M učiteli.

5. Zásady SGL – vytváření dotazů

5.1. Úvod do SQL – popis a vlastnosti

SQL - *Structured Query Language* (Strukturovaný dotazovací jazyk) - je obecný nástroj pro manipulaci, správu a organizování dat uložených v databázi počítače. Je v prvé řadě určen uživatelům, i když jej v mnoha směrech využívají i tvůrci aplikací. Je adaptovatelný pro jakékoliv prostředí.

Název tohoto nástroje je sice stručný, není však výstižný. SQL totiž není pouze dotazovací jazyk, s jeho pomocí je možno také definovat data, tedy strukturu tabulky, naplňovat sloupce tabulky (pole záznamů) daty a definovat vztahy a organizaci mezi položkami dat. Dále umožňuje řízení přístupu k datům, tedy udělování a odebírání přístupových oprávnění na různých úrovních, čímž chrání data před náhodným nebo úmyslným zničením, neautorizovaným čtením nebo manipulaci s nimi, dále také umožňuje sdílené využívání dat a zajišťuje hladký průběh činností, přistupuje-li k datům více uživatelů současně.

SQL je specializovaný programovací jazyk, který se používá ve vhodném prostředí buď uživatelsky nebo interaktivně k okamžitému řešení úloh (nejčastěji dotazy), nebo se jeho příkazy vkládají do hostitelského jazyka. SQL není však plnohodnotným samostatným programovacím jazykem, např. proto, že se v něm ve většině implementací nenachází řídicí programové konstrukce a další požadované prvky, které by měl obsahovat každý obecný programovací jazyk. SQL je tedy standardizovaný nástroj pro práci s relačními databázemi. Nepředstavuje databázový systém, ale různě integrovanou součást systému řízení bází dat.

SQL je především interaktivní dotazovací jazyk - umožňuje získat odpovědi i na velmi komplikované dotazy téměř ihned. Je nástrojem neprocedurálním, s množinovým přístupem k datům a je jazykem standardizovaným, je srozumitelný, protože chápe data v podobě tabulek, což je snadno pochopitelné i uživatelům. Pracuje s relačními databázemi, ve kterých se uživatel dívá na data v podobě soustavy provázaných tabulek. Každá tabulka představuje množinu dat, která je uspořádaná v řádcích (záznamech) a

sloupcích (položkách). Na hodnotu dat se uživatel odkazuje jako na prvek v matici.

V převážné většině případů je výsledkem úlohy popsané v SQL nějaká množina dat z jedné nebo více tabulek, tzv. tabulka výsledků, která nemusí být vždy konečným produktem. Může sloužit jako množina vstupních údajů pro další zpracování. Např. pro výtisk etiket nebo vykreslení grafu, atd.

SQL může sloužit jako "společná řeč", zejména při provozu v sítích, na kterých se používají různé databázové produkty.

Jazyk SQL se používá také k vytváření náhledů (dotazů). Náhledy SQL umožňují vytvořit pro různé uživatele různé pohledy na struktury tabulek a na data. Každý uživatel tak vidí pouze ta data, která má vidět. Přitom data vidí uživatel opět v podobě jednoduché tabulky, i když ve skutečnosti data pocházejí z různých tabulek. Data zobrazovaná v náhledech jsou dynamická, tzn. změní-li se data v tabulkách (databázových souborech), změní se také data, které zobrazuje náhled. Také naopak. Pracuje-li se s aktualizačním náhledem (speciální typ náhledu, který umožňuje nejen data prohlížet, ale i přidávat a aktualizovat) a změní se v něm data, promítnou se změny do patřičných tabulek (databázových souborů).

Klíčovým pojmem v jazyku SQL je ***příkaz***. Každý příkaz začíná klíčovým slovem. Slovo vyjadřuje orientačně, jakou činnost daný příkaz provádí. Za klíčovým slovem následuje jedna nebo více volitelných klauzulí, které blíže specifikují povahu vykonávané činnosti, nebo určují data, s nimiž má příkaz pracovat.

Každá klauzule začíná klíčovým slovem, jako jsou např. FROM nebo WHERE. Některé klauzule jsou povinné, jiné volitelné. Každá implementace SQL používá kromě standardních klauzulí, daných konvencemi ANSI/ISO, své vlastní klauzule, někdy se i činnost standardních klauzulí mírně či více liší.

Jména objektů v jazyku SQL mají ve standardu délku 1-18 znaků, z toho první znak musí být písmeno, a jméno nesmí obsahovat mezery nebo interpunkční znaky. Při udávání jména je někdy potřeba jméno kvalifikovat, vyskytují-li se stejná jména, a není jednoznačně jasné, na který objekt se jméno odkazuje. Kvalifikace jména se provádí pomocí kvalifikátoru ´.´ (tečka). Často se kvalifikace používá se jmény tabulek:

```
<Owner>. <Table_name>
```

nebo při kvalifikaci sloupců, což se v databázových systémech používá velmi často:

```
<Tablename>. <Columnname>
```

obecně se v SQL připouští i kvalif. víceúrovňová:

```
<Owner>. <Table_name>. <Column_name>
```

5.2. Databáze SQL a dotazy

Úvodní do příkazů v SQL databázích

Základem je představit na jednoduchých příkladech základní příkazy jazyka SQL. Informace pak můžeme použít při programování ve Visual Basicu, Delphi, programování webových stránek. Základní úvod co je databáze je zmíněn v článku databáze úvod. Pokud již teorií databázi rozumíte, můžete se podívat jak vytvořit databázi v programu Microsoft Access.

Seznam SQL příkazů

Mezi nejdůležitější příkazy patří: SELECT, INSERT, DELETE, CREATE, FROM, WHERE a mnohé další v následujících kapitolách si na příkladech jednotlivé příkazy představíme. Nejprve si ale uděláme jednoduchou tabulku, na které si příkazy představíme (to aby bylo jasnější, co který ten příkaz dělá)

Vytvoření testovací tabulky

zamestanci : Tabulka

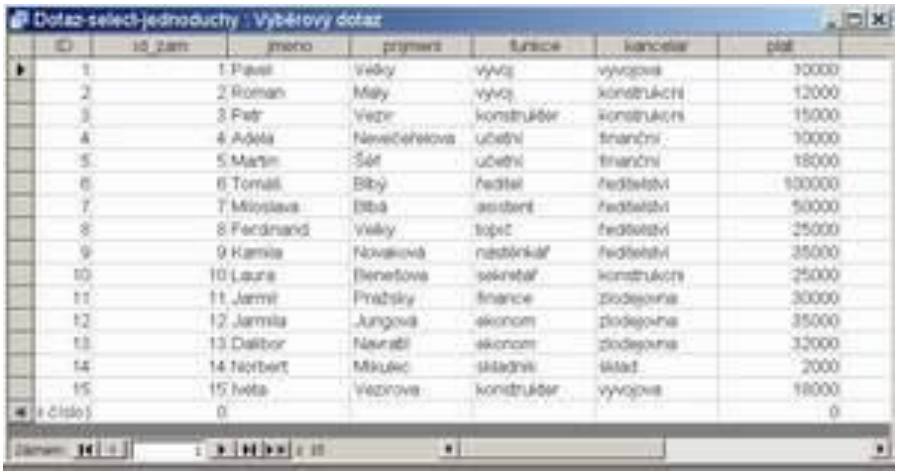
ID	id_zam	jmeno	prijmeni	funkce	kancelar	plat	vek
1	1	Pavel	Velky	vyvoj	vyvojova	10000	20
2	2	Roman	Maly	vyvoj	konstrukcni	12000	25
3	3	Petr	Vezir	konstrukter	konstrukcni	15000	35
4	4	Adela	Nevečeřelova	učetní	finanční	10000	20
5	5	Martin	Šéf	učetní	finanční	18000	23
6	6	Tomáš	Bílý	ředitel	ředitelství	100000	35
7	7	Miloslava	Bílá	asistent	ředitelství	50000	16
8	8	Ferdinand	Velky	topič	ředitelství	25000	66
9	9	Kamila	Novaková	nástěnkář	ředitelství	35000	25
10	10	Laura	Benešova	sekretař	konstrukcni	25000	55
11	11	Jarmil	Pražsky	finance	zlodejovna	30000	36
12	12	Jarmila	Jungová	ekonom	zlodejovna	35000	28
13	13	Dalibor	Navratil	ekonom	zlodejovna	32000	19
14	14	Norbert	Mikulec	skladnik	sklad	2000	48
15	15	Iveta	Vezirova	konstrukter	vyvojova	18000	47
(Nové číslo)	0					0	0

Záznam: 1 z 15

Abychom si ujasnili pojmy z SQL jazyky, je vhodné si to testovat na nějaké jednoduché tabulce. Poté, co již je plně chápeme, je můžeme nasadit na tabulku s 10.000 řádky. Tuto tabulku můžeme vytvořit v MySQL, FoxPro, Access či nějakém jiném databázovém programu.

Příkaz SELECT

Důležitý příkaz, Použijeme naši vytvořenou tabulku a otestujeme si na ní aplikace tohoto příkazu. Nejjednodušší forma příkazu je tato:



```
SELECT *
FROM employees;
```

Vypíše obsah celé tabulky "zamestananci"

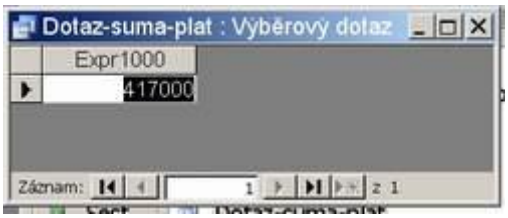

```
SELECT the name
FROM employees;
```

Vypíše obsah sloupce "jmeno" z tabulky "zamestananci"


```
SELECT DISTINCT function
FROM employees;
```

Vypíše obsah sloupce "funkce" z tabulky zamestanci, ale zobrazi jen položky bez duplicit (takže i když vyvojaře máme 4x vypíše se jen jednou)


```
SELECT AVG(plat)
FROM zamestanci;

SELECT SUM (flat)
FROM employees;
```

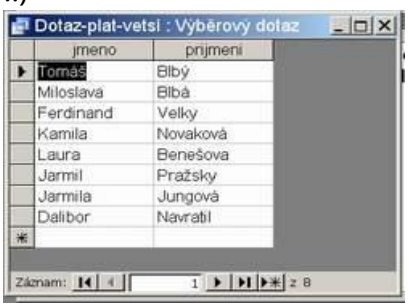


AVG(sloupec) vypočte půměr z daného sloupce, SUM(sloupec) vypočte sumu z hodnot v daném sloupci, případně můžeme i v samotném dotazu provádět výpočty.

```
SELECT SUM (PLAT + 500)
FROM employees;
WHERE
```

WHERE

Až budete mít 10.000 řádků, budete chtít omezit výpočty podle nějakých kritérií. Toho lze docílit příkazem WHERE. Spolu s použitím nějakých matematických symbolů (<,>,<=,>=, ..)




```
SELECT name
FROM employees
WHERE pay> 20000;
```

Vypíše jména zaměstnanců, jejichž plat přesahuje 20 000 Kč

Obecná syntaxe jednoduchého dotazu

SELECT *seznam sloupců*
FROM *tabulka*
WHERE *vyhledávací podmínky*
ORDER BY *sloupce řazení*

Seznam sloupců je buď seznam sloupců oddělených čárkou, nebo znak * pro výběr všech sloupců.

Název sloupce je buď pouze název sloupce nebo **název_tabulky.název_sloupce**. Také je možné použít zápis **tabulka.*** pro výběr všech sloupců z tabulky.

Tabulka **může být cokoliv z:**

- relace (reálná tabulka v databázi)
- výsledek jiného SELECT dotazu
- pohled
- výsledek operátoru JOIN (virtuální tabulky).

Vyhledávací podmínka je podmínka, které musí být splněna pro každý záznam, který se vypíše ve výsledcích dotazu. Samozřejmě pokud chceme nějaký záznam vypsat, tak musí nejprve existovat ve zdrojové tabulce. Část WHERE je tedy *omezující* podmínka, pokud není uvedena, tak se vypisují všechny řádky z tabulky.

Sloupce řazení je seznam sloupců oddělených čárkou, podle kterých bude výsledná tabulka seřazena, první sloupec je primární kritérium řazení, druhý sloupec je sekundární řazení - pokud nelze rozhodnout podle prvního sloupce, ...

Příklad 1

Zadání: Vyberte z databáze všechny Radky a Tomáše.

Řešení 1

```
SELECT name, surname, nickname FROM person
WHERE name = 'Radek' ORname = 'Tomas'
```

SELECT jmeno, prijmeni, rezdivka **FROM** osoby **WHERE** jmeno='Radek' **OR**jmeno='Tomáš'

Můžeme použít běžných logických operátorů **AND** a **OR**, na většině databázových serverů se dají zapisovat také jako **&&** a **||**. Pro porovnání řetězců se dá použít obyčejné =, je však potřeba myslet na několik věcí. Podle nastavení serveru záleží nebo nezáleží na velikosti písmen, výchozí nastavení (a tedy častější) je, že na velikosti písmen nezáleží. Na serveru Akela.mendelu.cz na velikosti písmen záleží. Dále podle nastavení collation serveru (a databáze) záleží nebo nezáleží na diakritice. Na serveru Akela.mendelu.cz opět na diakritice záleží, čili jméno je nutné psát přesně tak jak je uložené v databázi.

Řešení 2

SELECT jmeno, prijmeni, prezdivka **FROM** osoby **WHERE** jmeno **IN**('Radek','Tomáš')

Alternativa k předchozímu dotazu je použití operátoru **IN**. IN je množinový operátor, který testuje, zda se hodnota nachází v zadané množině, tedy pro každý řádek se vezme hodnota sloupce *jmeno* a testuje se, jestli je v množině ('Radek', 'Tomáš'). Množina může být definovaná buď přímým výčtem hodnot, nebo dalším SQL dotazem.

Příklad 2

Zadání: Vyberte z databáze všechny osoby, které bydlí v Praze.

Řešení – krok 1

```
SELECT id_address, FROM FROM WHERE address mesto = ’Praha’
```

SELECT id_adresy, mesto **FROM** adresy **WHERE** mesto ='Praha'

V prvním kroku zjistíme všechny adresy, které jsou v Praze. Nyní je vhodné připomenout si **schéma databáze**, ze kterého zjistíme, že v tabulce osoby existuje sloupec id_adresy, který se váže na sloupec id_adresy v tabulce adresy. Úvaha je tedy taková, že pokud vybereme všechny osoby, jejichž adresa se nachází ve výsledku výše uvedeného dotazu, tak dostaneme právě ty osoby, jejichž adresa je v Praze.

Řešení – krok 2

```
SELECT name, receive, nickname FROM people
WHERE id_addresses IN ()

SELECT DISTINCT id_address FROM WHERE address
mesto = ’Praha’ ()
```

SELECT jmeno,prijmeni,prezdivka **FROM** osoby **WHERE** id_adresy **IN**(

SELECT DISTINCT id_adresy **FROM** adresy **WHERE** mesto ='Praha'()

Nyní jsou v dotazu dvě konstrukce SELECT. Druhý select je tzv. *pod-dotaz* a je to lehounce upravený dotaz vytvořený v prvním kroku. První úprava se týká toho, že nyní vybírá pouze sloupec id_adresy. To je zásadní, vzhledem k tomu, že nadřazený SQL dotaz hledá id_adresy, tak pod-dotaz musí vracet pouze id_adresy. Kromě toho operátor

IN pracuje pouze se skalární množinou a tedy pod-dotaz (pro definování množiny pro operátor IN) musí vždy vracet právě jeden sloupec. Pokud v pod-dotazu použijete více sloupců tak DB server přivítá chybou:ERROR: subquery has too many columns. Druhá změna spočívá v přidání klíčového slova **DISTINCT**, které zajistí, že vrácené hodnoty budou unikátní (viz následující příklad). V tomto konkrétním případě je zbytečné (protože id_adresy je vždy unikátní), ale vzhledem k tomu, že IN se může chovat velmi prapodivně s pod-dotazem, ve kterém se hodnoty opakují (ona to pak koneckonců není množina), tak jako preventivní univerzální opatření je DISTINCT dobrý.

6. SQL – složitější dotazy

6.1. Cvičení – dotazy SQL s více tabulkami

Příklad 1

Zadání: napište SQL dotaz, který vybere z databáze všechny osoby, které mají ICQ číslo; vyberte jméno, příjmení a ICQ číslo osoby; seřaďte podle příjmení vzestupně.

Varianta 1

Nejjednodušší - počítá s tím, že v tabulce typy_kontaktu, hodnota id,_typy_kontaktu = 1 odpovídá typu ICQ. Operátor JOIN pracuje vždy se dvěma tabulkami. Spojovací podmínka je vždy rovnost hodnot v nějakých sloupcích. Spojovací podmínka musí obsahovat obě tabulky, které se vyskytují v části JOIN. Spojovací podmínka musí obsahovat sloupce, které zajišťují vazbu mezi tabulkami. V tabulce osoby není žádný odkaz na tabulku kontakty. V tabulce kontakty je jeden odkaz na tabulku osoby - sloupec id_osoby. INNER JOIN vybere z tabulky pouze ty záznamy, které vyhovují spojovací podmínce - tedy pouze osoby, které mají nějaký kontakt (protože je však v dotazu zadaná i vyhledávací podmínka je v tomto konkrétním případě možné použít i LEFT i RIGHT JOIN, je to ovšem jen shoda okolností).

```
SELECT person.name, person.name, contact.contact FROM
     INNER JOIN contacts
          ON contacts.id_osoby = person.id_osoby
WHERE contacts.id_types_contact = ’1’
ORDER BY by ASC
```

Varianta 2

Stejná jako předchozí, ale využívá USING. Pokud je spojovací podmínka rovnost stejně pojmenovaných sloupců (id_osoby) lze její zadání zjednodušit.

```
SELECT person.name, person.name, contact.contact FROM
     Persons JOIN contacts USING (id_persons)
WHERE contacts.id_types_contact = ’1’
ORDER BY by ASC
```

Varianta 3

Lepší - vybere kontakty podle názvu typu kontaktu, nespoléhá se tedy na nějakou hodnotu id_typu_kontaktu. Název typu kontaktu je v tabulce typy_kontaktu. Je důležité myslet na to, že výsledkem spojení tabulek je tabulka a že operátor JOIN pracuje vždy se dvěma tabulkami.

```
SELECT person.name, person.name, contact.contact FROM
      (INNER JOIN contacts
            ON contacts.id_osoby = person.id_osoby)
                  INNER JOIN contact_types
                        ON contact_types.id_type_contact =
                        contacts.id_types_contact
WHERE contact_name.nazev = 'icq'
ORDER BY by ASC
```

SELECT je opět příkaz, který může mít spoustu dalších slov, kombinovat několik tabulek do sebe atd. Ale úplný základ sytaxe je: SELECT, pak následuje seznam sloupců, které chceme získat, případně agregační či jiné funkce, FROM a název tabulky, ze které chceme data získat, WHERE a výraz s omezeními, která musí platit pro žádané řádky. V našem případě chceme, aby sloupec mesto obsahoval přesně hodnotu "Chomutov".

S agregační funkcí to vypadá třeba takto – chceme zjistit kolik řádků, tedy lidí je z Chomutova:

```
SELECT COUNT (*) FROM lide WHERE mesto = "Chomutov";
```

Vrátí nám to jeden řádek v jednom sloupci a bude obsahovat číslo 2.

Podmínky můžou být mnohem složitější s boolean operátory AND a OR, závorkami a dokonce SQL funkcemi, jejichž seznam je opět v dokumentaci. Vyhledání lidí z Chomutova starší 30 let by se udělalo takto:

```
SELECT * FROM lide WHERE mesto = "Chomutov" AND age> 30;
```

Hvězdička znamená "všechny sloupce".

Napojení dalších tabulek do SELECT dotazu se dělá pomocí JOIN. To téma vyžaduje mít

víc než jednu tabulku a nějakou relaci mezi nimi. Například tabulka lide jak to máme, kde každý člověk má unikátní id a pak třeba tabulku nápady se sloupci clovek_id a nápad. Ve sloupci clovek_id by byly cisla lidi, kteří daný nápad vymysleli a ve sloupci nápad by byl text nápadu. Pak bychom mohli udělat dotaz, který by vypsal všechny nápady a ke každému přidal i sloupec s názvem člověka. Detaily jsou ale nad rámec tohoto úvodu:

```
SELECT napady.napad, lide.jmeno FROM napady
LEFT JOIN lide ON lide.id = napady.clovek_id;
```

Výsledkem bude tabulka nápadů, kde v prvním sloupci bude nápad a v druhém jméno člověka, který nápad vymyslel.

Když místo "SELECT sloupce" zadáme "DELETE", máme dotaz pro smazání řádků. Funguje podobně jako SELECT (je tam podmínka WHERE) ale místo získání výsledků smaže řádky odpovídající podmínce. Pro smazání Adama stačí zadat:

```
DELETE FROM lide WHERE name = "Adam";
```

Z těch nejpoužívanějších SQL dotazů ještě UPDATE. Ten provede úpravu řádku. Pokud Kateřina zestárne o jeden rok, provedeme aktualizaci třeba takto:

```
UPDATE people SET age = 30 WHERE name = "Catherine";
```

můžeme ale také použít matematický výraz a reference na existující hodnoty sloupců. Takže zvýšit hodnotu sloupce vek o jedna jde i takto:

```
UPDATE people SET age = age + 1 WHERE name = "Catherine";
```

To jsou všechny nejčastěji používané SQL dotazy (klauzule). Jednotlivé dotazy mohou mít víc slov pro složitější a přesnější dotazy. Pro jejich pochopení je však nutno studovat víc jednotlivé dokumentace, nebo hledat konkrétní problém. Například když použijeme v SELECT agregační funkci, získáme výsledek agregace celé tabulky. Pokud ale třeba chceme mít průměry teplot v rocích a máme teploty po měsících, musíme použít GROUP BY a když dáme do Google **group by years datetime mysql**, zjistíme, že potřebujeme DATETIME funkci YEAR:

```

```

```
GROUP BY YEAR (record_date)
```

Kromě GROUP BY má SELECT i ORDER BY, kde si zadáme podle jakých sloupců výsledky řadit a jestli sestupně nebo vzestupně. Důležité je někdy pořadí slov. Pro SELECT je opět v dokumentaci a vidíme, že nejdřív v dotazu musíme mít GROUP BY … a pak ORDER BY … .

7. Grafové databáze

Graf je datová struktura skládající se z vrcholů a hran. Grafová databáze je systém na ukládání a zpracování dat v podobě grafu. V mnoha případech je modelování domény grafem velmi přirozené – mezi takové domény patří vztahy mezi lidmi, mezi geny a proteiny, mobilní sítě, distribuční sítě různého druhu, neuronové sítě nebo třeba ekologické sítě zachycující interakce organismů.

Mezi grafové databáze se často řadí různé systémy, které spravují data v podobě grafů. Na základě takovéto definice by bylo teoreticky možné pohlížet i na relační databáze, resp. na jejich nadstavby jako na grafové databáze. Relační databáze ovšem neumožňují efektivní uložení a dotazování grafových dat.

7.1. Relační databáze

Průchod grafem v relační databázi vyžaduje joiny, a je tedy neefektivní pro průchod do hloubky. Jeden join za použití indexu vyžaduje logaritmický čas O(log n), bez použití indexu je to O(n log n).

Například vztahy mezi lidmi můžeme modelovat pomocí dvou tabulek: Člověk a Vztah. Pro každý průchod vztahem mezi dvěma lidmi obecně potřebujeme jeden join. Tradiční relační databáze jsou optimalizovány pro jednotky joinů. Jakmile počet joinů přejde do desítek, dostávají se relační databáze do potíží i přes použití indexů.

7.2. Opravdové databáze

Opravdovými grafovými databázemi nazveme pro účel tohoto článku takové databáze, které vyžadují pouze konstantní čas O(1) pro průchod hranou grafu.

Opravdové grafové databáze, podobně jako jiné NoSQL databáze, ukládají data v denormalizované (již „zjoinované“) podobě. U opravdové grafové databáze se tedy platí především u zápisu, zatímco dotazování (čtení) je levnější.

Příkladem opravdové grafové databáze je systém Neo4j, který je ve vývoji již více než deset let a je dostatečně vyspělý pro produkční nasazení, nebo například novější systém OrientDB.

Příkladem databází, které jsou schopné ukládat a dotazovat data v podobě grafů, ale ne nutně efektivně procházet grafy, je většina triplestorů (Sesame, Jena, Virtuoso) nebo FlockDB (projekt Twitteru).

Podobně jako je tomu u většiny NoSQL databází, ani pro grafové databáze neexistuje jednotný dotazovací jazyk. Nicméně mnoho grafových databází implementuje jazyk na procházení grafů Gremlin vytvořený ve spolupráci s autory Neo4j.

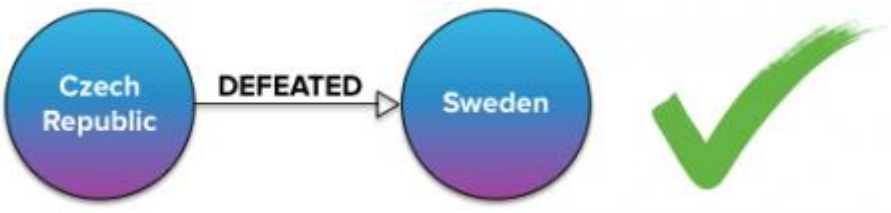
Přechod z relačního světa do světa grafů vyžaduje posun v uvažování a pohledu na data.

I když grafy jsou často mnohem intuitivnější než tabulky, postřehl jsem stále opakující se chyby u lidí, kteří s modelováním grafů začínají. V tomto článku se podíváme na jednu z nejčastějších chyb – modelování obousměrných vztahů, a nakonec si ukážeme reálný příklad, ve kterém budeme nadále v seriálu pokračovat.

7.3. Jednosměrné vztahy

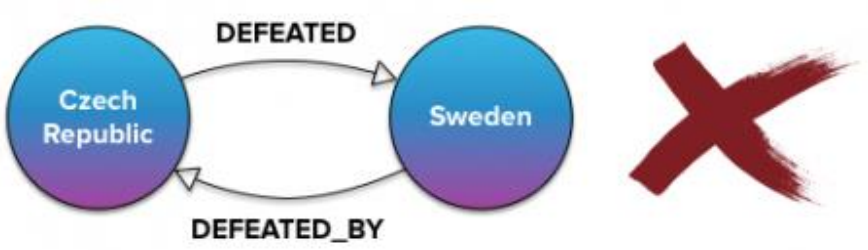
Vztahy v Neo4j musí být nějakého typu, který dává vztahu sémantický význam a také musí mít určený směr. Právě ten vyjadřuje smysl mezi entitami. Jinými slovy, vztah je nejednoznačný bez určení směru vztahu.

Například následující graf ukazuje, že Česká republika porazila (DEFEATED) Švédsko v ledním hokeji. Kdyby se směr vztahu obrátil, Švédové by byli mnohem šťastnější. Bez směru vůbec nevíme, kdo je vítězem, a proto je takový vztah nejednoznačný.



Všimněte si, že z existence tohoto vztahu vyplývá vztah opačném směru, jak je ukázáno níže v dalším grafu. Jedná se o velmi častý případ. Ještě jednou si popíšeme na jiném příkladu, že film Pulp Fiction režíroval (DIRECTED) Quentin Tarantino znamená také, že Quentin Tarantino je režisérem (IS_DIRECTOR_OF) filmu Pulp Fiction. A tímto by mohlo dojít k velmi mnoho párovým vtahům.

Všimněte si, že z existence tohoto vztahu vyplývá vztah v opačném směru, jak je ukázáno níže v dalším grafu. Jedná se o velmi častý případ. Ještě jednou si popíšeme na jiném příkladu, že film Pulp Fiction režíroval (DIRECTED) Quentin Tarantino znamená také, že Quentin Tarantino je režisérem (IS_DIRECTOR_OF) filmu Pulp Fiction. A tímto by mohlo dojít k velmi mnoho párovým vtahům.



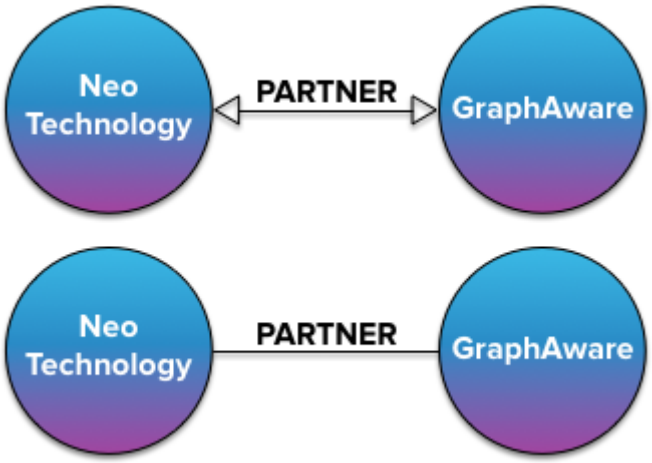
Právě této chyby se lidé často dopouštějí při modelování grafu v Neo4j a vytváří obous-

měrné vztahy. Vzhledem k tomu, že jeden vztah vyjadřuje druhý (symetrická relace), je to nehospodárné jak z hlediska prostoru, tak z průchodu (traverzování) grafem. Neo4j umožňuje procházet vztahy v obou směrech, takzvaně i proti směru hrany. Navíc díky způsobu, jakým Neo4j data ukládá, nezávisí rychlost průchodu vztahem na jeho směru.

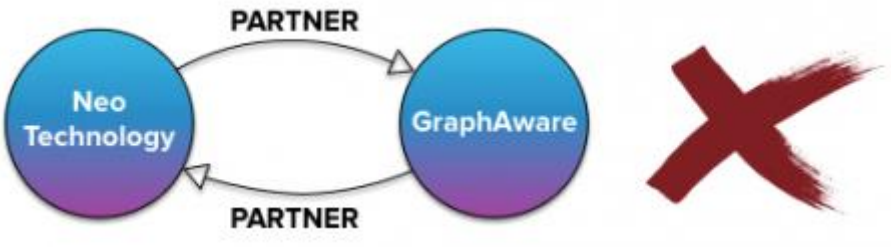
7.4. Obousměrné vztahy

Některé vztahy jsou přirozeně obousměrné. Klasickým příkladem je Facebook nebo vztah přátelství. Vztah je vzájemný – když s někým přátelíte, tak se on přátelí s vámi (možná tedy). V závislosti na tom, jak se díváme na model grafu, bychom o takovém vztahu mohli říci, že je obousměrný neboli neorientovaný.

Příkladem GraphAware a NeoTechnology jsou partnerské společnosti. Jelikož se jedná o vzájemný vztah, mohli bychom modelovat obousměrný neboli neorientovaný vztah.



Ale právě toto není možné v Neo4j – každý vztah musí mít počáteční a koncový uzel. Začátečníci se tak často uchylují k následujícímu modelu, který trpí přesně stejným problémem jako výše zmíněný model ledního hokeje s nadbytečným vztahem.



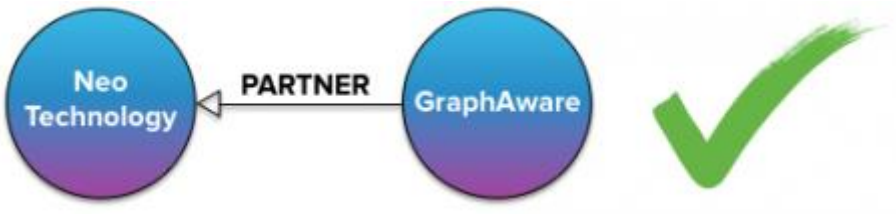
Neo4j API umožňuje vývojářům zcela ignorovat směrovost vztahů při psaní dotazů, pokud si to přejí. Například v jazyce Cypher by vyhledání všech partnerských společností s NeoTechnology mohlo vypadat takto:

```
MATCH (neo) - [: PARTNER] - (partner)
```

Výsledek by byl stejný jako sloučení výsledků z těchto dvou různých dotazů:

```
MATCH (neo) - [: PARTNER] -> (partner) and MATCH (neo)
<- [: PARTNER]
```

Proto je správný (nebo alespoň nejúčinnější) způsob modelování partnerských vztahů pomocí jediného PARTNER vztahu s určením libovolného směru.



7.5. Shrnutí

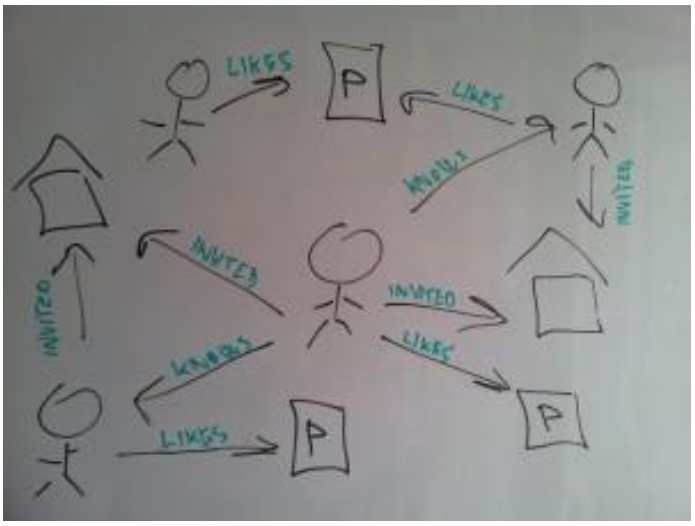
Vztahy v Neo4j lze procházet v obou směrech se stejnou rychlostí. Navíc směrovost může být zcela ignorována. Z tohoto důvodu není třeba vytvářet dva různé vztahy mezi entitami, pokud mají hrany opačného směru stejný význam.

7.6. Metodika modelování grafové databáze

V následujících článcích si ukážeme návrh, implementaci a testování grafové databáze Neo4j na projektu BestPartyToday (jedná se celosvětový pártylist, který nabízí detailní statistiky k akcím). V současné době se v projektu používá databáze MySQL, která obsahuje desítky milionů záznamů, takže nás také čeká přesunutí dat z relačního databázového úložiště do grafového. Ovšem v dnešním příspěvku si pouze namodelujeme podobu grafové databáze z vybraných tabulek a atributů současné struktury MySQL databáze.

O konceptuálním návrhu grafové databáze se pojednává jako o tzv. Whiteboard friendliness. Pro rychlé promítnutí všech myšlenek a postřehů, které byly vzneseny během návrhového brainstormingu, postačí pouze flipchart. Výsledný nákres je velmi jednoduché následně prezentovat ostatním projektovým skupinám (retail department, account managers, sales managers atd.), které nemají technické vzdělání. Pochopení návrhu pro ně už není překážkou, jelikož nákres je možné jednoduše přizpůsobit do podoby reálného života.

7.7. Realizace konceptuálního modelu



Na fotografii finálního návrhu grafové databáze pro projekt BestPartyToday je využito pěti nejvýznamnějších tabulek současné relační databáze, které jsou reprezentovány ikonami znázorňující nodes a směrovými šipkami prorelationships. Skrze entity (nodes a relationships) budeme procházet graf a získávat požadovaná data, která budou zpracována a zobrazena uživateli aplikace. Jejich význam je popsán níže v tabulce. Přehled entit grafového modelu.

8. NoSQL databáze

NoSQL databáze NoSQL databáze jsou tématem této práce, proto je jim věnována největší část. Nejdříve je třeba říct, co to vlastně NoSQL databáze jsou. Dále je kapitola věnována CAP teorému, na kterém je vysvětleno, že jedním z důvodů, proč existuje tolik NoSQL produktů, je to, že každý z nich může zajistit jen určité vlastnosti na úkor jiných a uživatel se pak musí rozhodnout, která kombinace vlastností je pro něj nejdůležitější a podle toho vybrat produkt. Dále se práce věnuje škálovatelnosti, tedy schopnosti navyšování výkonu databázového 15 serveru, ať již samotným vylepšováním serveru o výkonnější hardware nebo rozprostřením databáze na více serverů. Spolu se škálovatelností je třeba se taky zmínit o shardingu, tedy o technice, která rozděluje databáze na více částí, jež mohou fungovat samostatně a rychleji, než jeden celek. Poté následuje seznámení s nejznámějšími NoSQL produkty, kde jsou následně vybráni tři zástupci, se kterými je provedeno bližší seznámení. Popis Jak již zaznělo v úvodu, NoSQL databáze jsou ty, které nejdou relační cestou řízení dat, ale jinou. Nejsou primárně postavené na tabulkách a nevyužívají SQL pro práci s daty. Jejich doména je vysoká optimalizace pro vyhledávání, na úkor malé funkcionality, která se často omezuje na prosté ukládání dat. Tyto nedostatky, v porovnání s SQL, jsou kompenzovány škálovatelností a výkonem u určitých datových modelů. NoSQL se hodí na ukládání velkého objemu dat, u kterých není třeba uchovávat vzájemné vztahy. Strukturovaná data jsou ovšem povolena. U transakcí nemohou NoSQL databáze nabídnout plnou podporu ACID, nýbrž pouze Eventual consistency (Výsledná shoda). Jde o situaci, kdy díky zanedbání ACID získáme větší dostupnost a také lepší škálovatelnost. Tento přístup bez „silné konzistence“ je vhodný pro NoSQL, které ho taky často aplikují. NoSQL má distribuovanou architekturu, která je odolná vůči chybám. Některá data bývají umístěna na několika serverech a tím selhání jednoho se dá tolerovat. Tyto databáze většinou škálují horizontálně a spravují velké objemy dat, u kterých je důležitější výkon než konzistence.

NoSQL je doslovně kombinací dvou slov: No a SQL. Je to tedy technologie, která stojí proti SQL. Zkratka může být matoucí a mezi lidmi neexistuje jednotný názor na to, co znamená. Nejvíce rozšířený je ovšem ten, který tvrdí, že jde o acronym „Not only SQL.“ Ať už zkratka znamená cokoliv, NoSQL je dnes zastřešující název pro všechny databáze, které nejdou cestou známého RDBMS (relační systém řízení báze dat), ale jdou svojí vlastní, často spojovanou s velkými objemy dat.

8.1. Co je NoSQL?

Především je důležité zdůraznit, co NoSQL není – rozhodně to není *NO SQL*, tedy *trendy* odmítání relačních databází. Zkratka NoSQL znamená Not Only SQL, tedy uvědomění si toho, že relační databáze není jediná možnost řešení persistence, že existují i alternativy, které mohou být v některých případech vhodnější.

U rozsáhlejších aplikací pak můžeme dojít k tomu, že na některá data je vhodná relační databáze, na jiná NoSQL databáze a na další data úplně jiná NoSQL databáze. Tento pragmatický přístup, kdy se mixuje použití více databází v jednom projektu, se často označuje jako **polyglot persistence**.

NoSQL databáze vznikaly (a vznikají) jako řešení reálných problémů – není to tedy bláznivý výmysl akademiků odtržených od reality, ale vysoce praktická záležitost. Zrod mnoha NoSQL databází je spjat s projekty, které se musely vypořádat s obrovským množstvím dat – Facebook (**Cassandra**), Google (**BigTable**), Amazon (**Dynamo**), LinkedIN (**Voldemort**) atd.

Použití NoSQL databáze může mít ale smysl i tehdy, pokud máme méně dat (které by v pohodě zvádla relační databáze) – třeba protože nějaká NoSQL databáze nabízí datový model, který je pro naši aplikaci přirozenější.

Ale zpět k úvodní otázce. NoSQL databáze je software pro perzistenci dat, který je alternativou ke klasickým relačním databázím (nic objevného). NoSQL databází existuje mnoho a dají se rozdělit z **mnoha hledisek**. Ve velmi specifických případech může dávat smysl vyvinout i vlastní NoSQL databázi (ale myslete na **NIH**).

8.2. Kdy použít NoSQL

Rozhodnutí, zda použít místo ověřené relační databáze neznámou NoSQL databázi, není jednoduché, a hlavně to není rozhodnutí jen technické, ale i ekonomické.

NoSQL databáze např. často nemají takové možnosti dotazování jako relační databáze. Zkuste se třeba zamyslet nad tím, jak by ovlivnilo vaši aplikaci, kdyby jedinou podporovanou DB operací bylo uložení řádku a jeho načtení podle nějakého ID (nepřeháním!). To vede k tomu, že je na vaši aplikaci přenášena odpovědnost (např. generování sekundárních indexů), kterou jste dosud považovali za samozřejmou součást databázového enginu. Použití NoSQL databáze tak často znamená kompletní redesign celé aplikace, takže hodně učení a práce pro aplikační programátory (kteří nepracují zadarmo).

Většina dnešních programátorů vyrůstala ve světě, kde jedinou možností uložení dat byla relační databáze. Umí tak s ní dobře pracovat (v čemž jim pomáhají léty vyladěné tooly), jsou na ni schopni napasovat jakýkoliv model a už při úvodním seznamování s novým projektem jim v hlavě automaticky naskakují tabulky a vazby mezi nimi. Takže

stojí za zvážení, zda se opravdu vyplatí investovat čas (peníze) do učení diametrálně odlišných technologií a zda nemůže být ekonomičtější použít léty ověřené relační databáze namísto nové NoSQL databáze.

Tím nechci říct, že se má absolutně kašlat na vzdělávání a inovaci, ale situace na trhu práce je tristní a ne každý má to štěstí (já ano :-)), že pracuje s bandou nadšených vývojářů, kteří čtou blogy, zajímají se o novinky v oboru a po večerech si zkouší nové technologie. Mnohdy člověk musí pracovat s lidmi, pro které je programování jen způsob obživy, odpracují si svých 8 hodin a tím to pro ně končí. A dělat s takovýmto týmem projekt, který používá NoSQL databázi, může být nákladné.

Náklady spojené s přechodem na NoSQL databázi se ale týkají celého týmu. Také vaši admini si budou muset umět poradit s novým SW v infrastruktuře, naučit se ho spravovat, zálohovat, tunit atp.

Co jsem chtěl říct tímhle zamyšlením? NoSQL je zajímavá alternativa k zavedeným relačním databázím a rozhodně má své místo na trhu a proto doporučuji se o NoSQL databáze zajímat a nějaké si zkusit osahat. Pokud si proti nějaké NoSQL databázi zkusíte napsat jednoduchou aplikaci, neuvěřitelně vám to rozšíří obzory.

Nepoužívejte ale NoSQL databáze jen protože je to cool, protože by to *mohlo* být lepší nebo protože je používá konkurence. Řiďte se rozumem, ne emocemi. Dobře si spočítejte, co vám NoSQL přinese a jaké budou náklady. Pokud se už pro NoSQL rozhodnete, tak myslete na to, že se nemusíte úplně vzdávat *pohodlné* relační databáze – v projektu nemusíte používat jednu jedinou databázi…

NoSQL databáze - Systém databází s možností horizontálního i vertikálního škálování. NoSQL databázový systém nevyužívá tabulky, jako je tomu u relačních databází, v tomto případě je cílem jednoduchost vzhledu a možnost horizontálního i vertikálního škálování. NoSQL je optimalizovanou databází, kdy jsou využívány klíče i hodnoty. Struktura ukládání dat je u NoSQL rovněž odlišná, kupříkladu se jedná o struktury stromovou nebo grafovou. NoSQL databázové systémy získávají na stále větší popularitě, především pokud hovoříme o segmentu real-time web. Dosud ale nejsou tolik rozšířené, což je dáno především absencí podpory transakčního modelu ACID či neúplnou standardizací rozhraní. Navíc jsou pro některé firmy NoSQL databáze dosti nákladné.

NoSQL

NoSQL je databázový koncept, ve kterém **datové úložiště** i zpracování dat používají jiné prostředky než tabulková schémata tradiční **relační databáze**. Motivací k tomuto přístupu mohou být jednoduchost designu, horizontální i vertikální **škálovatelnost** a jemnější kontrola dostupnosti. Databáze bez SQL jsou často vysoce optimalizovaná úložiště typu klíč-hodnota (ne vždy). Díky odlišné struktuře ukládání dat (např. stromová,

grafová) oproti **RDBMS**, je i **algoritmická složitost** pro různé operace odlišná. Obecně se vhodnost aplikace daného typu databáze liší podle řešeného problému.

Segment NoSQL databází v současnosti významně roste a prospívá především v oblasti **big data** a **real-time webu**. NoSQL systémům se také občas říká „nejen SQL“ pro zdůraznění faktu, že často umožňují dotazy v **SQL** (či podobném) jazyce. V kontextu **CAP teorému** NoSQL úložiště často potlačují konzistenci ku prospěchu dostupnosti a tolerance k narušení sítě.

Bariéry k rozsáhlejšímu nasazení těchto úložišť do praxe jsou např. nepřítomnost plnohodnotné podpory transakčního modelu **ACID**, použití (různých) nízkoúrovňových dotazovacích jazyků, nedostatečná standardizace rozhraní a vysoké realizované investice podniků do SQL v minulosti.

9. Transakce

Transakce je logická jednotka práce sestávající z jednoho nebo více SQL příkazů, které jsou atomické z hlediska zotavení se z chyb. SQL transakce automaticky začíná SQL inicializačním příkazem (SELECT, INSERT). Změny, které realizuje jedna transakce, nejsou viditelné pro ostatní konkurenčně probíhající transakce, pokud daná transakce neskončí.

Transakce může skončit jedním ze čtyř způsobů: – COMMIT končí transakci úspěšně a změny jsou trvale zaznamenány – ROLLBACK přeruší transakci a všechny změny se anulují, databáze se vrátí do stavu před transakcí – v rámci programu - skončí-li program úspěšně, končí úspěšně i SQL transakce – v rámci programu - končí-li program chybou, zruší se transakce SQL.

Řízení přístupu do databáze SQL definuje dva příkazy na řízení přístupu k tabulkám: GRANT a REVOKE Bezpečnostní mechanizmus je založený na – autorizačních identifikátorech – vlastnictví – privilegiích.

9.1. Řešení: Transakce

Transakce lze chápat jako sekvence databázových operací, které splňují ACID:

- **Atomicita** – Všechny operace v rámci transakce se provedou buď všechny nebo žádná.
- **Konzistence** – Před i po vykonání transakce se databáze nachází v konzistentním stavu. Musí být splněna všechna integritní omezení.
- **Izolace** – Jednotlivé transakce jsou izolované. Změny prováděné v průběhu zpracovávání dané transakce nejsou viditelné v ostatních transakcích.
- **Trvalost (durability)** – Po úspěšném ukončení transakce jsou data uložena a nemohou být ztracena.

9.2. Transakce v SQL

V jazyce SQL máme pro práci s transakcemi k dispozici následující příkazy:

BEGIN	spouští transakci ...příkazy v rámci transakce...
COMMIT	uloží a ukončí transakci
BEGIN	...příkazy v rámci transakce...
ROLLBACK	vrátí změny provedené transakcí a ukončí ji

Úrovně izolace

Požadavky ACID jsou relativně silné a jejich zajištění je časově náročné. Proto je možné tyto podmínky zeslabit pomocí tzv. úrovní izolace. Rozeznáváme 4 úrovně (od nejslabší po nejsilnější):

1. READ UNCOMMITTED

Transakci je umožněno číst data zapisovaná jinou transakcí, aniž by tato jiná transakce byla ukončená pomocí COMMIT. Čtení takových dat označujeme jako tzv. **dirty read**. Takto přečtená data mohou být nekonzistentní (čas plyne shora dolů):

2. READ COMMITTED

Na této úrovni izolace již nemůže dojít k **dirty read**. Data, která přečteme, byla vždy zapsána transakcí, která byla úspěšně ukončena pomocí COMMIT. Avšak může dojít k tzv. **non-repeatable read**. Pokud v rámci transakace čteme nějaká data vícekrát, může se stát, že se při opakovaném čtení pokaždé nevrátí stejný výsledek. Mezi čteními dat naší transakcí totiž mohla být úspěšně ukončena souběžná transakce, která zapsala/upravila/smazala nějaká data.

3. REPEATABLE READ

Na této úrovni je zajištěno, že nedojde k non-repeatable read. Data, která jednou přečteme, se již nemohou při dalším čtení změnit. Může však nastat tzv. phantom read. Pokud v rámci transakce čteme nějaká data vícekrát, může se stát, že výsledek opakovaného čtení bude obsahovat nové řádky, které při předchozím čtení ve výsledku nebyly. Mezi čteními dat naší transakcí totiž mohla být úspěšně ukončena souběžná transakce, která zapsala nová data.

4. SERIALIZABLE

Nemůže dojít k žádnému výše uvedenému fenoménu, vynucuje ACID.

10. Procedury a funkce, triggery a sekvence

Funkce, metody a procedury – jaký je mezi nimi rozdíl?

Všechny tři (funkce, metody i procedury) jsou si poměrně dost podobné – je to POJMENOVANÁ posloupnost příkazů, část programu, kterou můžeme opakovaně volat z jiných částí programu.

10.1. Formální pohled

Z formálního hlediska se řídíme názvosloví daného programovacího jazyka.

- **Funkce** je pojem z funkcionálního programování, vyskytuje se v jazycích jako je JavaScript nebo třeba Haskell.

- Jako **metody** označujeme funkce v objektově orientovaném programování (OOP), kde dochází k propojení datových struktur a funkčního kódu do objektů. Metody nalezneme v jazycích jako Java, Smalltalk a dalších.

- **Procedura** je pojem známý z procedurálních jazyků. Nalezneme je např. v Pascalu, ale také v některých databázových systémech – tzv. uložené procedury.

Na funkce, metody a procedury se ale můžeme dívat z věcného hlediska (podle jejich vlastností a smyslu) a odchýlit se od terminologie konkrétního programovacího jazyka.

Funkce

Funkce v programování má velmi blízko k funkcím, jak je chápeme v matematice. Funkce má určitý definiční obor (typ vstupních parametrů) a určitý obor hodnot (typ návratové hodnoty).

Příklad jednoduché funkce v JavaScriptu:

```
Function sum (a, b) {return a + b};
```

V objektových jazycích jako Java máme sice metody a ne funkce, to nám ale nebrání v nich funkce psát:

```
Public static int sum (int a, int b) {return a + b};
```

Přestože sečti() – sum() je formálně metoda (veřejná a statická), ve skutečnosti se jedná o

funkci a třída zde plní pouze úlohu jmenného prostoru (společně s názvem balíčku) a nevytváříme její instance (resp. nemusíme). Jedná se o návrhový vzor Knihovní třída.

V Javě najdeme takové funkce např. ve třídě java.lang.Math. Následující funkce vrací větší ze dvou čísel:

```
Int x = java.lang.Math.max (a, b);
```

Procedura

Procedura je zvláštním případem funkce – NEMÁ návratovou hodnotu a nemusí mít ani vstupní parametry. Používají se často při dávkovém zpracování – např. každou hodinu zavoláme proceduru, která zpracuje objednávky, které se nashromáždily v databázi, a předá je do jiného systému.

Příklad velmi jednoduché procedury v PostgreSQL, která sečte u dosud nesečtených řádků hodnoty sloupců a a b a výsledek uloží do sloupce c:

```
CREATE OR REPLACE FUNCTION add ()
RETURNS void AS
$ BODY $
UPDATE table of summaries
SET c = a + b
WHERE c is NULL
$ BODY $
LANGUAGE sql VOLATILE;
```

Tato procedura je napsaná v jazyce SQL – kromě toho můžeme psát plpgsql, který nám umožní tvořit i velmi složité procedury, případně můžeme použít jazyk C či jiný.

Výše uvedená procedura (formálně funkce) nemá žádné vstupní parametry ani návratovou hodnotu, je to prostě jen posloupnost příkazů jazyka, kterou jsme si nějak pojmenovali a uložili.

Zajímavostí je, že procedury můžou mít parametry – a to nejen klasické vstupní, ale i výstupní (nebo vstupně/výstupní). Procedura s výstupním parametrem nám slouží podobně jako funkce – přestože nemá klasickou návratovou hodnotu. Proceduře můžeme např. předat nějaká data a ona nám je upraví.

Takové procedury si můžeme simulovat i v jazyce Java. Mějme nějakou přepravku (strukturu):

```
Public class Person {public String name;
Public String Surname; Public String fullname};
```

A proceduru:

```
Public class StoredProcedures {public void reportNameName
(Person o) {o.celeName = o.name + ”” + oName}};
```

Proceduře předáme data (osobu) a dojde k jejich požadované úpravě. Ovšem pokud programujete tímto stylem v Javě, pravděpodobně děláte něco špatně a připravujete se o hlavní výhody OOP.

10.2. Metoda

Metody jsou součástí třídy a (pokud nejsou statické) jsou úzce spjaté s danou instancí třídy (objektem). Nejsou to tedy funkce pracující s nějakými globálními proměnnými a daty, ale mají přístup k proměnným dané instance (v tomto případě osoby).

Předchozí příklad můžeme přepsat „objektovějším“ způsobem:

```
Public class Person {public String name; Public
String Surname; Public String getCelé Name ()
{return name + ”” + surname}};
```

Třídu s uloženými procedurami vůbec nepotřebujeme a požadovanou funkcionalitu přesuneme blíže k datům (do třídy Osoba). Všimněte si, že ani NEMUSÍME ukládat vypočtenou hodnotu celéJméno – vypočteme ji až ve chvíli, kdy ji někdo bude potřebovat, tzn. zavolá metodu get CeléJméno().

Co je trigger?

(Trigger je procedura, která je automaticky spouštěna, když se něco stane. Mohu ho definovat na DML (modifikace) i DDL (vytváření) operace. A navíc mohu říct, jestli se má provést před, nebo po dané operaci. (BEFORE, AFTER) Zajímavostí je, že trigger se může jmenovat stejně jako tabulka (avšak není to doporučováno) a že lze definovat libovolný

počet triggerů na stejnou tabulku i událost. Avšak pozor: nelze určit pořadí triggerů, ve kterém budou volány a triggery nesmí na sobě vzájemně záviset. V triggerech se také nesmí využívat rekurzivní volání. V těle triggeru lze provádět libovolné SQL dotazy - INSERT, UPDATE, DELETE. Avšak pokud chcete použít dotaz SELECT, musíte ho využít v kombinaci s klíčovým slovem INTO, které zajistí uložení načtených dat do lokálních proměnných, nebo kurzoru.

Typy triggerů

- DML triggery na tabulkách
- INSTEAD OF triggery na pohledech (views)
- Systémové triggery na DATABASE or SCHEMA: DATABASE - triggery se spustí na každou událost pro všechny uživatele, SCHEMA - triggery se spustí na každou událost pro zvoleného uživatele

Trigger je nástroj, který zajišťuje automatické provedení programu v jazyce SQL při vložení, zrušení nebo změně záznamu v určité tabulce. Trigger může například zajistit, že:

- před vložením nového záznamu do tabulky bude provedena kontrola jeho obsahu a doplněny hodnoty některých sloupců;
- po smazání záznamu se provedou následné akce;
- při změně hodnoty určitého sloupce v záznamu se porovná stará a nová hodnota a na základě jejich vztahu se provedou další akce.

Triggery zjednodušují tvorbu aplikací, protože přenášejí část práce databázové aplikace na server. Umožňují centralizované definování pravidel platných pro informační systém. Existuje-li například v podnikovém informačním systému tabulka zaměstnanců, lze pomocí triggerů popsat, jaké všechny akce musí být provedeny při přijetí nebo propuštění zaměstnance, změně platu nebo přeřazení do jiného oddělení. Tyto akce se naprogramují na jednom místě, ale budou sloužit všem aplikacím, které manipulují s tabulkou zaměstnanců. Dodržení pravidel pro údržbu evidence zaměstnanců pak bude zajišťovat server automaticky a konzistence dat bude zajištěna bez ohledu na možné chyby v měnících se aplikacích.

Trigger je pojmenovaným objektem patřícím do databázové aplikace. Má tyto vlastnosti:

- je svázán s určitou tabulkou a reaguje na změny v této tabulce;
- reaguje na právě jednu z SQL akcí **INSERT**, **DELETE** nebo **UPDATE**, triggery reagující na UPDATE mohou navíc **specifikovat množinu sloupců**, jejichž změna je spouští; trigger se spouští také prováděním obdobných akcí při práci s formuláři;
- provádí se buď před (BEFORE) provedením výše specifikované akce, nebo po ní

(AFTER);

- pokud akce ovlivňuje více než jeden záznam, pak se může spouštět buď pro každý ovlivněný (tj. vložený, zrušený nebo změněný) záznam zvlášť, nebo pro všechny záznamy najednou;

- může obsahovat podmínku, která se vyhodnotí před provedením triggeru a pokud není splněna, trigger nebude spuštěn;

- specifikuje akci (zapsanou v jazyce SQL), která bude automaticky provedena, jakmile jsou splněny podmínky pro spuštění triggeru.

Při provedení konkrétní akce mohou být splněny podmínky pro spuštění více než jednoho triggeru. V takovém případě jsou spuštěny po řadě všechny. V současné podobně návrhu normy SQL 3 není zahrnut způsob, jak by uživatel v textu triggerů mohl definovat pořadí jejich spouštění. Je pravděpodobné, že před definitivní redakcí normy bude tento způsob specifikován a poté bude přidán i do 602SQL.

Provádění triggerů může měnit obsah databáze a tím spouštět další triggery. Spouštění triggerů se může do sebe libovolně zanořovat.

Při provádění triggerů se nekontrolují práva (triggery se provádějí s maximálními právy). Aby díky tomu nemohlo dojít k neoprávněnému přístupu k datům, může triggery vytvářet pouze uživatel obsazený do **role** Author v aplikaci, do níž trigger patří.
V **zamčené aplikaci** není tedy možné triggery vytvářet.

INSERT a UPDATE triggery se nespouštějí při importu dat do tabulky. DELETE triggery se nespouštějí při smazání tabulky jako objektu (při mazání záznamů samozřejmě ano). Spouštění triggerů při akcích vyvolaných aktivní referenční integritou závisí na nastavení **příznaku kompatibility** SQLOPT_NO_REFINT_TRIGGERS (4096)

Pro účely ladění může být výhodné **trasovat** spouštění triggerů
(situace TRACE_TRIGGER_EXEC), triggery lze také **debugovat** stejně jako procedury.

Triggery a transakce

Trigger se vždy provádí jako součást **transakce**, v níž je spuštěn příkaz, který trigger spustil. **Chyba**, která případně nastane při provádění triggeru, má stejné důsledky, jako chyba ve spouštěcím příkazu - trigger nelze odvolat bez odvolání tohoto příkazu, příkaz nelze odvolat bez odvolání změn provedených triggerem.

Chyby **kategorie** "rollback exception condition", pokud se vyskytnou uvnitř triggeru, ignorují vnitřní deklarace handlerů (tj. handlery v triggeru a procedurách volaných tímto triggerem). Zachytit je lze pouze handlerem pro spouštěcí příkaz.

Uvnitř triggerů se nesmí provádět žádné explicitní **transakční příkazy**, v opačném případě nastane chybový **sqlstate** 2D000 (SQ_INVAL_TRANS_TERM).

11. Analytické nástroje OLAP

11.1. OLAP – informace pod kontrolou

Kteří zákazníci přinášejí 80 % zisku? Jak se změnila úspěšnost udržení zákazníků po zavedení věrnostního programu? Jak různé události ve firmě (kampaně, zavádění nových produktů, změny firemních procesů apod.) a na trhu (aktivity konkurence, změny ekonomických podmínek apod.) ovlivňují kolísání prodeje? Jaké je srovnání s minulým obdobím? Tyto a podobné otázky si klade každý realistický manažer. Každý den se snaží pochopit chování firmy na základě analýzy údajů, které má k dispozici. V dobře fungujících společnostech jsou součástí vybraných oddělení analytici, kteří tuto roli přebírají a manažerům poskytují pouze zdůvodněná doporučení. Všichni tito lidé potřebují mít připravenou infrastrukturu a nástroje, které jim umožňují tuto pro firmu velmi potřebnou činnost vykonávat.

11.2. Co a jak zjednodušuje práci?

Cesta k úspěšné analýze dat začíná už při jejich přípravě. Prvním důležitým krokem je shromáždění dat z podnikových systémů do formy vhodné pro analýzu a reportování. Již při této přípravě je nutné dbát na správné sjednocení různorodých dat a ošetření jejich kvality. Musíme vždy důvěryhodně zdokumentovat vazby na zdrojové systémy, aby výsledky byly vždy věrohodné a doložitelné - pro případ řešení chyb, které se v původních datech vyskytují. Výsledná data se ukládají do relační databáze, kde jsou k dispozici v rozsahu potřebném k operativním analýzám. Takto relačně připravená data slouží k vytváření detailních reportů o výkonnostních parametrech společnosti a jednodušším analýzám založeným na detailních datech. V následujícím zjednodušeném příkladu si však představme požadavek na jiný typ informací: Firma s celostátní působností prodává různé kategorie výrobků. Manažer člení jednotlivé pobočky do regionů. Kategorie prodávaných výrobků mají tři úrovně - kategorie, podkategorie, konkrétní výrobek. Firma má logicky definováno, že prodej se řídí plánem. Reálné plnění plánu je pak dáno součtem za všechny výrobky a všechny pobočky.

11.3. Jak tyto informace získat?

Pro tyto potřeby slouží pokročilé analýzy, které jsou označovány jako multidimenzionální. Manažer může sledovat měřitelné parametry firmy v souvislostech, různých úhlech a různých mírách detailu. K takové analýze je nutná technologie OLAP (on-line analytical processing), která zajistí uložení a předpočítání dat takovým způsobem, že následné

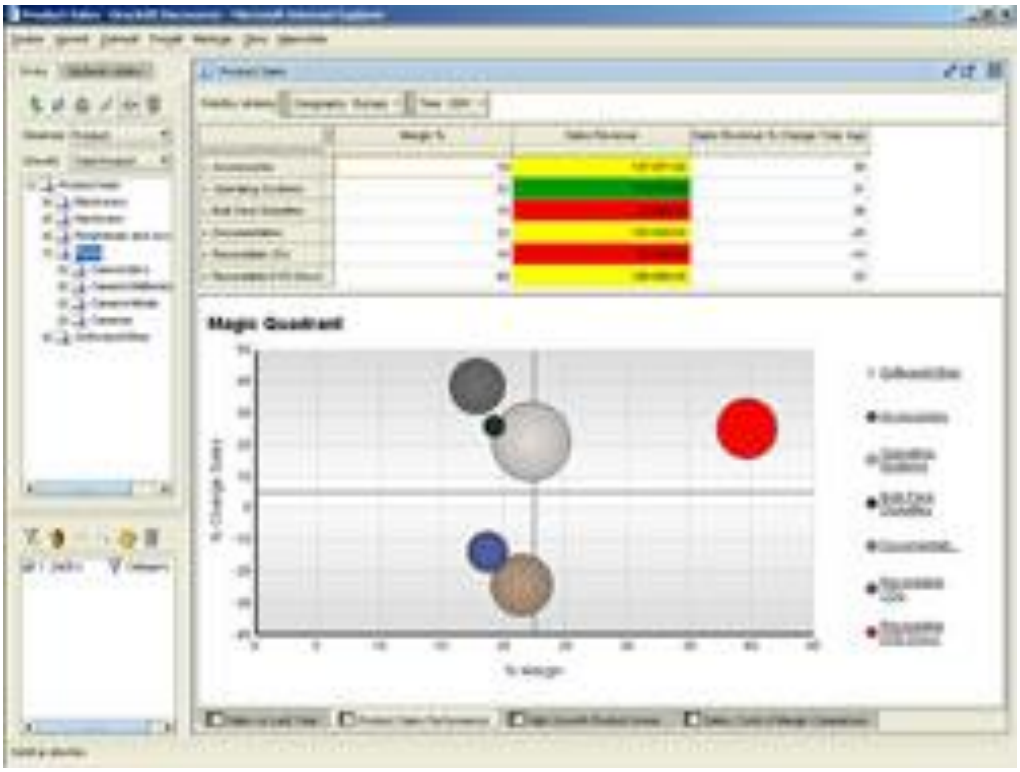
dotazy trvají přiměřenou dobu. Technologie OLAP přináší možnost pracovat s daty na

sumární úrovni, zde identifikovat problém či zajímavou oblast a tzv. drilováním postupovat k takové úrovni detailu, která je pro naše rozhodování zajímavá. Typickou oblastí, na které se výhody technologie OLAP demonstrují, je analýza prodeje.

Obr. 1: OLAP v nástroji Oracle Business Intelligence – Discoverer

Manažer může prostřednictvím nástrojů pro práci s OLAPem kontrolovat, jak se vyvíjí realita proti plánu. Pokud plán není plněn, trendová křivka včas tuto informaci ukazuje, a konec roku nemusí skončit fiaskem. Sumární čísla je možné snadno rozdrilovat o úroveň níže, pak jsou vidět sumární plnění plánu v jednotlivých regionech. Analýza může ukázat, že problém spočívá pouze v jediném z nich, ostatní plán plní. Dalším drilováním se dostane na úroveň poboček, kde identifikuje pět takových, které v problematickém regionu výrazně zaostávají. Systém umožní rozdrilování na úroveň kategorií a subkategorií až na jednotlivé výrobky, které se v daném místě nedaří prodávat. Na základě takové informace pak lze navrhnout a provádět korektivní kroky, které situaci zlepší. Z příkladu je patrné, že data při použití technologie OLAP jsou vlastně uložena ve stromové struktuře - kostce. Zde jsou pro každou úroveň hloubky tohoto stromu předpočítány hodnoty pro příslušný podstrom. Taková hierarchická struktura je definovatelná na většině podnikových dat. Výrazně zpřehledňuje a zjednodušuje analýzu při jejich vyhodnocování. Cenou za toto zjednodušení je čas a místo potřebné k vypočítání a uložení zmíněného stromu. Je-li však kostka naplněna, dokáže přesně odpovídat na velmi komplikované dotazy, zejména provádět srovnání časových řezů omezených složitými podmínkami. Typický komplikovaný dotaz zní: "Kterých deset zákazníků zaznamenalo nejvyšší nárůst obratu s mojí firmou ve srovnání s minulým rokem, a to v okresech, které patří mezi 20 % mých nejméně ziskových?" Výsledek pak může sloužit pro nadstandardní péči o tyto zákazníky. S technologií OLAP je možno

dotazovat data z mnoha úhlů pohledu - mnoha dimenzí - a tyto úhly libovolně parametrizovat. Jak jsme si ukázali, pro tyto typy dotazů není vhodné ukládat data v relační podobě.



Obr. 2: Analýza prodeje v nástroji Oracle Business Intelligence - Discoverer

11.4. Analytické nástroje

Až do tohoto okamžiku jsme popisovali způsoby a technologie uložení dat. K práci s nimi (provádění analýz) jsou zapotřebí vhodné nástroje. Volba vhodného analytického nástroje je pro efektivní získávání informací naprosto klíčovou záležitostí. Je nutné zohlednit zejména zkušenosti analytiků, kteří ve firmě již dnes pracují s prostředky, které mají k dispozici - velmi často jsou používány nástroje z rodiny Microsoft Office. Je proto důležité, aby zvolené řešení umožňovalo dostatečnou spolupráci s nástroji, se kterými jsou lidé zvyklí pracovat - to vede k nižším nákladům na adoptování efektivnějšího způsobu práce s informacemi. Zároveň musí být také zajištěna bezpečnost přístupu k informacím, automatizovaná distribuce reportů, kvalitní grafický výstup pro prezentování, případně integrace s firemním intranetem.

Hlavní jsou lidé

Přes záplavu technologií však nesmíme zapomenout na lidi. Jedněmi z nejcennějších lidí, které firma může získat, jsou zkušení analytici. Protože pouze zkušený analytik může vyčíst ze záplavy dat podstatné informace a z nich formulovat doporučení pro management.

11.5. OLAP kostka

Prohledávání do hloubky je úplný algoritmus (projde každý uzel). Jeho princip spočívá v tom, že ex OLAP krychle (online analytical processing) je způsob organizace dat, který rozšiřuje dvojrozměrně tabulkové uspořádání tak, že každá datová dimenze je uložena v jedné ose kostky. Tím překonává některá omezení relačních databází.

Uspořádání dat do vektorů kostek umožňuje k nim zpětně přistupovat z různých hledisek (dimenzí) stejným způsobem. Odpadá tím na výkonnost systému náročné spojování mnoha tabulek RDBMS. Nicméně fyzické ukládání dat do kostek neumožňuje rychlou editaci, v takovém případě je třeba přepracovat celou kostku. Kostka je tvořena hodnotami, které jsou kategorizovány do dimenzí. Struktura je implementována relačními tabulkami ve hvězdicovém schématu či schématu sněhové vločky. Jedná se typicky o rodič-potomek (parent-child) strukturu, kde rodičovské prvky reprezentují konsolidaci potomků a zároveň ony samy mohou být agregovány do svých rodičovských prvků.

Základní operace s datovými kostkami

Pro analytické potřeby se provádí následující operace s datovými kostkami, které mají za účel zpracování a projekci dat tak, aby usnadnily jejich pochopení.

Krájení kostky: omezení jedné nebo více dimenzí na podmnožinu o jednom prvku.

Kostkování: omezení jedné nebo více dimenzí na podmnožinu o dvou a více prvcích.

Roll up a drill down: jedná se o navigaci datovou hierarchií směrem nahoru a dolů.

Pivotování: otáčení kostky za účelem získání jiné perspektivy na vztahy dat.

Agregace: Konsolidace podle vztahů určených vzorci.

12. Specifika databázových systémů. Technologie přístupu k databázím. Geografické informační systémy.

Databáze (neboli datová základna, též databanka) je systém souborů s pevnou strukturou záznamů. Tyto soubory jsou mezi sebou navzájem propojeny pomocí klíčů. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim. Tento software se v české odborné literatuře nazývá systém řízení báze dat (SŘBD). Běžně se označením databáze – v závislosti na kontextu – myslí jak uložená data, tak i software (SŘBD).

12.1. Přehled přístupu k datům v technologii ASP.NET

Visual Studio 2010

Webové aplikace běžně přistupují ke zdrojům dat pro ukládání a načítání dynamických dat. Můžete psát kód pro přístup k datům pomocí tříd z oboru názvů System.Data (obecně označované jako technologie ADO.NET) a z oboru názvů System.Xml. Tento přístup byl běžný v předchozích verzích technologie ASP.NET.

Avšak technologie ASP.NET také umožňuje provádět vázání dat deklarativně. Pro většinu běžných scénářů není zapotřebí psát žádný kód, tyto mohou být:

- Výběr a zobrazení dat.
- Řazení, stránkování a také cachování dat.
- Aktualizace, vkládání a mazání dat.
- Filtrování dat pomocí parametrů za běhu.
- Vytvoření scénářů předloha-podrobnosti (master-detail) pomocí parametrů.

Technologie ASP.NET obsahuje několik typů serverových ovládacích prvků, které jsou součástí modelu deklarativního vázání dat, včetně ovládacích prvků zdroje dat, ovládacích prvků vázání dat a rozšiřujícího ovládacího prvku pro dotazy. Tyto ovládací prvky zvládají základní úlohy, které jsou vyžadovány modelem bezstavového webu s cílem zobrazit a aktualizovat data na webových stránkách technologie ASP.NET. Ovládací prvky umožňují přidat na stránku funkcionalitu vázání dat, aniž by bylo nutné porozumět podrobnostem životního cyklu požadavku stránky.

13. Databázové objekty

Pojem „databáze“ je často zjednodušován na to, co je ve skutečnosti databázový systém (databázový stroj) nebo též systém řízení báze dat. Ten neobsahuje pouze tabulky – ty jsou jedny z mnoha tzv. databázových objektů (někdy též databázových entit). Pokročilejší databázové systémy dále obsahují:

- pohledy neboli views – SQL příkazy, pojmenované a uložené v databázovém systému. Lze z nich vybírat (aplikovat na ně příkaz SELECT) jako na ostatní tabulky.
- indexy pro každou tabulku. Klíče jsou definovány nad jednotlivými sloupci tabulek (jeden klíč jich může zahrnovat i více) a jejich funkce je vést si v tabulkách rychlé LUT (look-up tables – „pořadníky“) na sloupce, nad nimiž byly definovány, vyloučit duplicitu v záznamech nebo zajišťovat fulltextové vyhledávání.
- spouště neboli Trigger– mechanizmus nad jednotlivými řádkami tabulky (případně samotnou tabulkou), který se vyvolá po změně, odstranění nebo přidání řádky, případně smazání tabulky a provede předprogramovanou akci (například kontrolu integrity dat, doplnění hodnot…)
- uživatelem definované procedury a funkce – některé databázové stroje podporují ukládání pojmenovaných kusů kódu, které provedou v databázi nad danými tabulkami určitou sekvenci příkazů (procedury) nebo navíc vrátí nějaký výsledek (uživatelské funkce). Mohou mít parametry, které se většinou dělí na vstupní (IN), výstupní (OUT) a vstupně-výstupní (INOUT).
- události, též (počeštěně) „eventy“ – de facto procedury, spouštěné v určitý (uživatelem definovaný) datum a čas nebo opakovaně s definovatelnou periodou. Mohou sloužit k údržbě, promazávání dočasných dat či kontrolování referenční integrity.
- formuláře – některé databázové systémy jako např. Microsoft Access umožňuje uživatelům vytvářet vstupní formuláře pro vizuálně přívětivé zadávání hodnot. Uživatel si může např. nadefinovat rozložení jednotlivých vstupních polí z dané tabulky, popisky atd.
- sestavy nebo též reporty – podobně jako u formulářů sestavy umožňují uživateli definovat layout s políčky dané tabulky, do kterého se při použití doplní aktuální hodnoty. Používají se pro výstup dat (tisk, prezentaci nebo pouhé zobrazení). Sestavy mohou být např. doplněny o filtry, které vyfiltrují jen kýžené záznamy.
- **uživatelská oprávnění** – u lepších databázových systémů je samozřejmostí nabídnout možnosti, jak oddělit jednotlivé úrovně přístupu k ostatním objektům databáze jejich uživatelům. Možností bývají desítky, s rozlišením na jednotlivé typy příkazů, které ten který uživatel bude nebo nebude mít oprávnění spustit.
- partitioning – způsob, jak rozdělit data v tabulce na více pevných disků a tím rozložit zátěž na ni kladenou
- procesy – databázové stroje umí podat přehled o procesech, které jejich služeb

aktuálně využívají.

- **proměnné nastavení** – typicky desítky proměnných, které lze přenastavovat a tím ovlivňovat chování a výkon databázového stroje jako takového.
- **collation** – MySQL má pokročilé možnosti pro nastavení několika desítek znakových sad a porovnávání, souhrnně nazývané collation. Nastavení collation může být provedeno na jednotlivé textové sloupce, celé tabulky i celé databáze (s kaskádovitou dědičností). Collation ovlivňuje i řazení, například hodnota utf8_czech_ci zajistí správné řazení podle češtiny (tedy včetně diakritiky a včetně ch).
- **vizuální E-R schéma** – (v MySQL INFORMATION.SCHEMA). Vizuální reprezentace vztahů (relací) na sobě závislých polí (cizích klíčů) mezi tabulkám.

13.1. Základní pojmy databázové technologie

Úvod k databázovým technologiím

S příchodem mikropočítačů a jejich vstupem do života každého z nás se zdá, že vzrůstá zájem o počítačově zpracovávaná data, a to ne pouze ve smyslu něco si vypočítat, ale především ve smyslu něco se dozvědět. Vysvětlení je zcela pochopitelné, uvážíme-li, že i ty nejdokonalejší počítačové hry časem omrzí a že programovat v Basicu výpočty typu řešení soustavy dvou rovnic o dvou neznámých nepřináší až tak velké uspokojení.

Člověk by si rád uspořádal a vyhledával některé informace, které často používá a které se v průběhu používání mění. Za předpokladu vhodných prostředků pro ukládání takových informací ve formě dat by byl ochoten prosedět několik večerů u klávesnice a příslušná data si sám vyrobit, koupit nebo vyměnit s jiným podobným nadšencem. Příkladem může být knihovna, úřad, zpracovávání letenek, městské muzeum, nemocnice, podnik apod.

Hotový objekt popsaného typu můžeme nazvat informačním systémem (IS). IS je tedy soubor prvků ve vzájemných informačních a procesních vztazích (informační procesy), které zpracovávají data a zabezpečují komunikaci informací mezi prvky. Z uživatelského hlediska by měl mít IS takové vlastnosti, aby manipulace s ním byla co nejjednodušší (vstup dat, formulace dotazů, použití aplikačních programů), a současně, aby funkce, které poskytuje, byly dostatečně silné a rychlé. Předložit uživateli pružný „prázdný“ IS lze dnes poměrně rychle, avšak s dostatečnou mírou znalostí.

Naším cílem bude ukázat si tzv. databázovou technologii zpracování dat. Databázová technologie je soubor pojmů, prostředků a technik sloužící pro vytváření informačních systémů (IS). Na nejzákladnější úrovni si můžeme představit architekturu IS s databází takto: data jsou organizována v databázi (DB), jsou řízena balíkem programů, který se nazývá systém řízení bází dat (SŘBD). Databáze a SŘBD tvoří tzv. databázový systém (DBS). V naší terminologii můžeme jednoduše psát DBS = DB + SŘBD. IS využívá data z DBS buď přímo, nebo je zpracovává aplikačními programy.

13.2. Geografický informační systém

(GIS; anglicky Geographic information system) je geografický informační systém, který umožňuje ukládat, spravovat a analyzovat prostorová data.

Většina objektů a jevů reálného světa se vyskytuje na některém místě zemského povrchu (např. strom, dům, řeka) nebo má vztah k některému místu na zemském povrchu (občan má někde trvalé bydliště, výrobek byl vyroben v určité továrně). Zároveň se tyto objekty vyskytují v daném prostoru společně s mnoha dalšími objekty a navzájem se ovlivňují. Proto znalost umístění a vzájemných prostorových souvislostí mezi objekty je velmi významná a může sehrát důležitou roli v řadě oborů lidské činnosti, od návrhu umístění jaderné elektrárny až po návrh obchodní sítě a vyhodnocování její úspěšnosti.

Prakticky to znamená, že v našich datech v počítači musíme mít zaznamenáno obojí současně, tj. jak vlastní údaje o objektu, tak údaje o jeho poloze. Tomuto typu dat říkáme geografická (nebo prostorová) data a počítačovému systému, který umožňuje ukládat a využívat taková data, říkáme geografický informační systém, zkráceně GIS.

Základní komponenty geografických informačních systémů

Základní komponenty geografických informačních systémů jak uvádí Vít Voženílek ve své knížce Geografické informační systém I. Pojetí, historie, základní komponenty (1998)[6] jsou:

- Hardware
- Software
- Data
- Obsluhující personál

Pro efektivní práci systémů je nezbytná jejich vyváženost. Hardware, neboli technické vybavení, představuje technickou základnu geografických informačních systémů. Software, neboli programové vybavení, představuje soubor programů vykonávající veškeré operace systému. Data jsou klíčovým prvkem každého geografického informačního systému. GIS je z pohledu organizační struktury skutečným systémem. Jeho fungování je souhrnem činností, které zabezpečují jednotlivé funkce systému.

13.3. Geografická data

Dimenze geoobjektů

Základním dělením geoobjektů je dělení podle počtu dimenzí. Reálné objekty na zemském povrchu jsou vždy trojrozměrné. Do prostředí GIS se však převádí (transformují) dle potřebné úrovně zjednodušení.

- 0D geoobjekty – bezrozměrné objekty, body, definované pouze svou polohou. Příkladem může být například autobusová zastávka v GISu modelujícím dopravu nebo GSM vysílač v GISu mobilního operátora modelující pokrytí signálem.
- 1D geoobjekty – jednorozměrné obj., úseky čar (hran, linií), s konečnou délkou a nulovou plochou. Pomocí 1D geoobjektů se nejčastěji modelují silnice, řeky apod.
- 2D geoobjekty – dvojrozměrné obj., mnohoúhelníky (plochy, polygony), s konečným obvodem a konečnou plochou.
- 3D geoobjekty – trojrozměrné obj., geometrické těleso. V GISech se používají výjimečně, ve specifických případech. Třetí rozměr je v GISech nejčastěji modelován pomocí tzv. digitálního modelu terénu (DMT, anglicky DEM) útvarem "Povrch" (surface) - spojenými topologickými plochami (2,5D).

Základem každého GIS jsou geografická data. Geografická data jsou informace o zemském povrchu a objektech, které se na něm nachází. Data si lze představit jako určité vrstvy informací, které se nazývají témata. Každé téma reprezentuje určitý prvek (např. silnice, jezera, města a podobně). Geografická data mohou být v GIS organizována dvěma základními modely:

13.4. Sběr dat

Získávání dat a přesun informací do systému zabere hodně času. Existuje celá řada metod používaných k zadávání dat do GIS, které jsou uloženy v digitálním formátu. Exitující data vytištěné na papíře nebo PET filmu mohou být digitalizovány nebo skenovány k výrobě digitálních dat. Digitizér produkuje vektorová data jako body, linie a polygony. Skenováním map v podobě rastru, který by mohly být dále zpracováván k výrobě vektorových dat.

Sbíraná data mohou přímo vstoupit do GIS použitím techniky zvané souřadnicová geometrie. Pozice z globálního družicového navigačního systému (GNSS) lze také shromažďovat a následně importovat do GIS. Současným trendem ve sběru dat umožňujícím uživatelům využívat počítače v terénu s možností upravovat živá data pomocí připojení k bezdrátové síti nebo i úpravy bez připojení k internetu. To eliminuje potřebu importovat a aktualizovat data v kanceláři poté, co byla shromážděna prací v terénu. To zahrnuje schopnost začlenit pozic získaných pomocí laserového dálkoměru. Nové technologie umožňují uživatelům vytvářet mapy, jakož i analýzy přímo v terénu, což činí projekty účinnější a mapování přesnější. Dálkově snímané data hrají také

důležitou roli při shromažďování údajů a skládají se ze senzorů připojených k platformě. Senzory zahrnují kamery, digitální skenery a LIDAR, zatímco platformy se obvykle skládají z letadel a satelitů. V Anglii v polovině roku 1990, hybridní balónky zvané Helikites první propagoval použití kompaktních digitálních fotoaparátů vzdušných výsadkových jako Geo-informační systémy. Software letadla měřící s přesností na 0,4 mm byl použit k propojení fotografií s měřením ze země. Helikitesy jsou levné a shromažďují přesnější údaje, než letadla. V poslední době vývoj miniaturních bezpilotních dronů. Například dron Aeryon Scout byla použit k mapování 50 akrového areálu s hustotou vzorkování každý 1 palec (2,54 cm) za pouhých 12 minut.

13.5. Multidimenzionální kostka

Základní stavební jednotkou v multidimenzionální databázi je kostka (cube, datová kostka, multidimenzionální kostka, hyperkostka). Kostka v multidimenzionální databázi se skládá ze sady dimenzí a měr. Dimenze (rozměr) kostky jsou kategorie, vůči kterým chceme data agregovat a analyzovat. Dimenze vznikají z tabulek relačních databází. Typickými dimenzemi v multidimenzionálních databázích jsou čas, poloha, výrobek. Dimenze se může skládat z řady úrovní, které dále zpřesňují údaje. Míry kostky jsou kvantitativní údaje, které chceme analyzovat. Tak jako dimenze jsou i míry odvozeny z tabulek relačních databází. Běžnými mírami jsou prodeje, výdaje, ceny, ale téměř každý kvantitativní údaj může být mírou multidimenzionální kostky.

13.6. Uložení dat v OLAP databázi

Zdrojová data z relační databáze mohou být uložena jednou ze tří základních metod: Datové kostky obsahují souhrnná data, které jsou základem multidimenzionální analýzy. Komplexní multidimenzionální aplikace (např. informetrie) obsahuje data související s různými kontexty, které ale mohou některé dimenze sdílet. Je proto srozumitelnější vytvářet jednu datovou kostku pro jeden kontext než používat datovou kostku zahrnující všechny kontexty. Datová kostka pozůstává jenom z těch dimenzionálních atributů, které sdílejí všechny její číselné atributy. To znamená, že dimenzionální atributy tvoří základ datové kostky. Pokud chceme získat informace z několika datových kostek, mohou být tyto atributy propojeny na úrovni jedné nebo více společných dimenzí. Navigaci mezi datovými kostkami uživatel nevidí.

Tabulky dimenzí popisují specifické vlastnosti dimenzí. Pro každý dimenzionální atribut může existovat nanejvýš jedna tabulka dimenze. Na rozdíl od původního hvězdicového modelu, v našem modelu mohou být dimenze datové kostky bez tabulek dimenzí. Na schématické úrovni se tabulka dimenze skládá ze jmen atributů, přičemž na úrovni příkladů pozůstává z hodnot těchto atributů.

V informetrii musejí být často souhrnná data analyzována na základě komplexních kritérií, které závisí na vlastnostech dimenzí. Aby bylo možné tato kritéria specifikovat,

měla by být informace použita ve více tabulkách dimenzí. Z toho vyplývá, že sémantické vztahy mezi tabulkami dimenzí se vytvářejí na základě sdílených hodnot některých atributů v dimenzionálních tabulkách.

INFORMAČNÍ A KOMUNIKAČNÍ TECHNOLOGIE

1. Didaktické prostředky, pomůcky, multimédia.

Informační a komunikační technologie, zkráceně ICT, (Information and Communication Technologies), česky též IKT, zahrnují veškeré informační technologie používané pro komunikaci a práci s informacemi.

Složky didaktické techniky:

- Jedná se o prostředky DT (přístroje a technická zařízení umožňující zprostředkování vizuálních, auditivních a audiovizuálních informací). Hlavním znakem je absolutní univerzalita.
- Učební pomůcky (jakékoliv vhodně zpracované učivo), velká specifičnost.

1.1. Didaktické pomůcky

1. **Originální předměty a reálné skutečnosti.**
   - Přírodniny v původním stavu (minerály, rostliny apod.), upravené (preparáty, vycpaniny, výbrusy, apod.).
   - Výrobky a výtvory v původním stavu (přístroje, umělecká díla apod.), upravené (sady a soubory vzorků, stroje v řezu apod.).
   - Jevy a děje, povahy fyzikální, chemické, biologické, sociální aj.
   - Zvuky, reálné zvuky, hlasové a hudební projevy.

2. **Zobrazení a znázornění předmětů a skutečností.**
   - Modely statické, funkční, stavebnicové, plošné apod.
   - Zobrazení prezentovaná přímo (obrazy, fotografie, diagramy aj.), prezentovaná prostřednictvím technických prostředků (staticky, dynamicky, interaktivně, virtuálně, 3D apod.).
   - Zvukové záznamy.

3. **Textové pomůcky tištěné či digitální.**
   - Knihy klasické, pracovní, programované, interaktivní.
   - Pracovní materiály, slovníky, tabulky, sbírky úloh, atlasy atd.

- Doplňková a pomocná literatura a informační zdroje.

4. **Pořady a programy prezentované (realizované) technickými prostředky.**
   - Pořady, výukové filmy, rozhlasové a televizní pořady apod.
   - Programy, informační, tutorské, repetiční aj.

5. **Speciální pomůcky.**
   - Experimentální soupravy, stavebnice aj.

1.2. Učební pomůcky

Existují 2 hlediska posouzení správnosti a vhodnosti zpracovaného učiva:

- ***Obsahové hledisko*** – posuzuje, zda učivo obsažené v pomůcce odpovídá současné úrovni poznané problematiky a je v souladu s cílem výuky daného typu školy - záležitost didaktiky daného vyučovaného předmětu.
- ***Formální hledisko*** – řeší otázku, zda lze bez problémů předložit žákům pomůcku daným prostředkem DT (velikost písma, vhodný rozměr a podoba atd.) – záležitost předmětu didaktická technika.

Pomůcka poskytuje informace:

- *obsahové* (týkají se předkládaného obsahu – pojmy a vztahy mezi nimi apod.),
- *interpretační* (ukazuje, jak s předloženými informacemi pracovat).

Pomůcka je velmi specifická – plně respektuje daný předmět, daný typ školy, ročník, téma i konkrétního učitele.

Význam a funkce učebních pomůcek a prostředků DT ve vyučování:

- napomáhají zvyšovat zájem žáků,
- podporují koncentraci žáků,
- podporují stálost pozornosti,
- motivační funkce,
- vedou jak k osvojení učiva, tak pracovních metod,
- rozšiřují informační možnosti v procesu vyučování a učení,
- umožňují zprostředkování těžko slovně sdělitelného učiva,
- umožňují věrnější poznání skutečnosti,
- spojují teorii s praxí.

Modernizace vyučovacího procesu:

Nejedná se o pouhé zavádění moderních prostředků DT do vyučovacího procesu, ale o následující sled fází.

- modernizace obsahu vyučovacího procesu,
- modernizace vyučovacích metod,
- zařazení moderních prostředků DT.

1.3. Multimédia

Jsou oblast informačních a komunikačních technologií, která je charakteristická sloučením audiovizuálních technických prostředků s počítači či dalšími zařízeními.

Jako multimediální systém se označuje souhrn technických prostředků (např. osobní počítač, zvuková karta, grafická karta nebo videokarta, kamera, mechanika CD-ROM nebo DVD, příslušný obslužný software a další), který je vhodný pro interaktivní audiovizuální prezentaci. Od počátku 90. let 20. století se začalo používat označení multimediální aplikace nebo multimediální software, které využívaly kombinace textových, obrazových, zvukových či animovaných nebo filmových dat.

2. Didaktická Technika

Prostředky didaktické techniky /DT/: přístroje a technická zařízení umožňující zprostředkování vizuálních, auditivních a audiovizuálních informací.

Předpoklady pro využití prostředků didaktické techniky /DT/:

- Důkladný rozbor vyučovací hodiny: stanovení struktury a těžiště hodiny.
- Vybereme poznatky, k jejichž objasnění jsou učební pomůcky nezbytné.
- Technickou formu pomůcky i její zařazení do vyučovací hodiny odvodíme z její funkce v procesu osvojování daného poznatku.
- Vyhodnocení vztahu mezi učební pomůckou použitou během vyučovací hodiny a materiály, které mají žáci k dispozici pro vlastní domácí přípravu (v učebnicích, sešitech, aj.) Postačí použití pomůcky v průběhu vyučovací hodiny nebo je vhodné, aby ji žáci měli k dispozici i pro domácí přípravu?

2.1. Multimediální didaktické prostředky

Multimédia - systémy technických prvků pro interaktivní audiovizuální prezentaci; umožňují uživateli práci s několika typy komunikačních prostředků (médií): s textem, obrazem, zvukem, počítačovou grafikou a animací, videem a s interaktivitou mezi systémem a jeho uživatelem. Tj. multimédia jsou charakteristická těmito znaky:

Textem? Audiem? Obrazem? Animací? Videem? Interaktivitou?

Interaktivita může zahrnovat různé možnosti:

- uživatel si vybírá obsah, kombinuje jeho části,
- uživatel ovlivňuje tempo, jímž multimediální systém prezentuje informace,
- uživatel si se systémem vyměňuje informace (např. formou otázky – odpovědi),
- uživatel řídí postup, jimiž systém prezentuje informace,
- uživatel vkládá do systému své části obsahu, porovnává je apod.

Multimediální didaktické prostředky – např.:

- výukový software (např. interaktivní multimediální DUMy), multimediální prezentace, multimediální záznamy výuky – např. videopřednášky, didaktické počítačové hry,
- systémy pro řízení studia (Learning Management System, LMS): plánování, tvorba, prezentace a řízení obsahu i vhodného prostředí, s nástroji pro práci LEKTORA i KLIENTA a jejich vzájemné komunikace (např. Moodle).

Rozdělení prostředků didaktické techniky (podle toho, na které lidské smysly při použití DT působíme):

- Prostředky vizuální techniky – působí na zrak.

- Prostředky auditivní techniky – působí na sluch,
- Prostředky audiovizuální techniky – působí současně na zrak i sluch.

Hypermediální a hypertextové didaktické prostředky:

Hypertext - text složený z bloků slov nebo symbolů elektronicky propojených cestami (elektronické linky) v otevřené a stále neukončené struktuře (síti) textů.

Hypermédium - digitální prostředek, který obsahuje aktivní odkazy nejen na texty, ale i tabulky, animace, obrazy, zvuk, video.

Hypertextové a hypermediální didaktické prostředky - digitální prostředky obsahující hypertextové a hypermediální prvky. Mají podobu sítě, v níž je hlavní linie textu propojena linky s texty a mediálními prvky, které hlavní text rozvíjejí, doplňují, znázorňují, ilustrují apod. Učící se subjekt postupuje v těchto sítích jedinečným, individuálním, nepředvídatelným způsobem.

3. Modelování. Definice. Teoretické modely.

Pojem „model“ se používá v různých kontextech. Tímto termínem se rozumí:

- matematické a mentální přiblížení daného problému,
- didaktické prostředky.

V hovorovém jazyku pojem model nejčastěji znamená:

- původní vzor něčeho vytvořeného;
- prototyp, např. model automobilu;
- ideální vzor, který chceme realizovat v určité činnosti nebo funkci, např. model perfektního učitele;
- strukturu toho, co je předmětem našeho zájmu, např. model školy;
- ve výtvarném umění se jedná o věc nebo člověka, který bude malován, tesán.

Ideální model je zobrazením zkoumaného jevu, které může obsahovat hypotetická vysvětlení a které může pomoct ověřit správnost dané hypotézy. Ideální modely, které jsou spojeny s konkrétní teorií, se nazývají teoretické modely. Materiální modely jsou existujícími předměty, jejichž vlastnosti umožňují rekonstruovat stavbu nebo podstatu zkoumaného předmětu nebo průběhu procesu.

Jedná se o zjednodušené a zkreslené reprodukce ideálních modelů, a proto je možné je uznat za „modely modelů“. Ještě než materiální model vznikne, musí existovat ve vědcově mysli jako určitá myšlenka, tedy jako ideální model. Z toho vyplývá, že modely jsou ve své prvotní formě abstraktními představami, které mohou být později ztvárněny konkrétním způsobem. Pro vědecké a vzdělávací účely nejdůležitější jsou ty modely, které jsou spojeny s vytvářením nových poznatků.

Teoretické modely

Mohou plnit tyto funkce:

- předávat klientům informace o teorii a jejím propojení s odpovídajícími experimentálními údaji;
- seznámit klienty s pojmy z oblasti dané teorie a rozvíjet schopnosti používat modely k řešení daných problémů;
- ověřit model experimentálním potvrzením nebo popřením předpokladů, jež se na něm zakládají, za účelem seznámení klientů s rozsahem a podmínkami jeho používání;
- verbalizovat model, což vede k formulování předpokladů příslušné teorie.

Modely můžeme rozdělit na statické a dynamické:

- ***Statické modely*** – většinou jsou rozkládací a zhotoveny v řezu perfektní názornost a rychlejší pochopení funkce.
- ***Dynamické modely*** – jsou to modely, které imitují pohyb a funkci.

Trenažéry – jsou to modely skutečných předmětů, zařízení aj., na kterých je možno se připravovat a zdokonalovat pro použití skutečných zařízení.

Virtuální počítačové modely – jsou to modely, které simulují některé jevy, tělesa, objekty v trojrozměrném či dvojrozměrném prostoru.

Funkce modelů, technických výukových prostředků ve výuce:

- *Funkce základní:* funkce informační, formativní, instrumentální.
- *Funkce didaktická:* funkce motivační a stimulační, racionalizační ve vztahu k učiteli i k žákům, zpevňovací opakováním informací, systemizační (začleňování informací do soustavy dříve získaných poznatků), funkce kontrolní a řídící.
- *Funkce ergonomické*: (nauka o vztazích mezi člověkem a pracovním prostředím a pracovními prostředky, usilujících o nejvýhodnější uspořádání pracovního prostředí) a řídící: např. snižování zbytečného času učitele i žáků, plné využití pro řízení výuky, regulace vlastního tempa učení podle dispozic a stavu psychiky.

PŘEDPOKLADY NOVÉHO, DYNAMICKÉHO POČÍTAČOVÉHO MODELU:

Prvním a nejdůležitějším předpokladem je, aby nově navržený model o mikrosvětě a jemu odpovídající vizuální učební pomůcky byly ***správné věcně (meritorně).*** Předpoklady nového dynamického počítačového modelu – vizualizace

4. Vzhled vizualizace

Pro zobrazení objektů ve výukovém procesu jsou v současnosti více využívány 3D interaktivní modely, jež nabízejí možnosti různých pohledů v různé míře detailnosti / na rozdíl od 2D vizualizace, která není dostatečně názorná a také maximálně nevystihuje realitu, současně se i někdy stává, že některé procesy, které jsou modelovány na pozadí, nechtěně splynou dohromady). 3D interaktivní modely nám přibližují reálnou skutečnost, kterou můžeme v reálném světě obtížné prohlížet a zkoumat, nebo již též neexistuje. Získané informace pak může klient asociovat s „prožitou" zkušeností, lépe si je zapamatovat a v případě potřeby „znovu vybavit". V obrazech se zrcadlíme my sami a naše zkušenost.

To v obraze neustále vytváří přitažlivou sílu a záhadu na emocionální úrovni, která podněcuje člověka k interaktivitě, sociální komunikaci a k tvorbě vlastních poznatků, poznatkových struktur a ke kritickému posuzování informací (Scruton, 2005).

Prostředky vizuální techniky

- pro nepromítané pomůcky

prostředek DT	pomůcka (nosič informace)
tabule	pedagogická kresba
magnetická tabule	papírové materiály, předměty na feritech
flanelová tabule	pomůcky podlepené flanelem nebo na suchém zipu
plexitová tabule	pedagogická kresba
flipchart	pedagogická kresba
stojan nebo háček na tabuli	nástěnná tabule, obraz nebo mapa

- pro statickou projekci

prostředek DT	pomůcka (nosič informace)
epiprojektor	pomůcka na neprůhledné podložce
diaprojektor	diapozitiv, diapás
zpětný projektor	pomůcka na průhledné podložce nebo velkoplošný diapozitiv

- pro dynamickou projekci

prostředek DT	pomůcka (nosič informace)
filmový projektor	němý film, filmová smyčka

Konstruktivismus a vizualizace

V posledních letech → implementace konstruktivismu jak do technického, tak i multimediálního, humanitního i cizojazyčného vzdělávání.

Konstruktivistické pojetí je označováno jako ideální pedagogické východisko pro vizualizaci, tedy pro zobrazování skutečnosti vnímané prostřednictvím zrakových receptorů.

Konstruktivistická pedagogika staví klienta do centra vzdělávacího procesu. Stejně tak vizualizace související s uplatňováním zásady názornosti předpokládá samostatného klienta, který dokáže částečně řídit a organizovat své učení.

Tradiční role lektora se přirozeně mění → stává se konstruktivistickým tutorem, facilitátorem a průvodcem.

Komunikaci mezi aktéry je třeba povzbuzovat a aktivovat jednak vytvořením příjemné atmosféry otevřeného prostoru pro sdílení názorů a jednak vhodnou koncepcí skupinové /kooperativní či kolaborativní/ práce.

Za hlavní znaky konstruktivisticky pojaté výuky můžeme považovat přechod od transmisivního vyučování, tzv. ,,tebeučení" k autoevaluaci, tedy sebeorganizaci, sebeřízení, sebeiniciaci, tzv. „ sebeučení".

Přístupy k médiím

- Mediální optimismus – optimistické přijetí médií, považující média za blahobytného hybatele společnosti. Transhumanismus, extropismus, singularitarianismus, techno-utopismus.
- Mediální pesimismus – Kritičtí oponenti mediálně optimistických směrů, poukazující na negativní aspekty technologicko-mediálního rozvoje, odmítající splynutí člověka a technologie (médií).
- Mediacismus – přílišné spoléhání se na média. Víra v to, že lidstvo bude schopno plně kontrolovat technologická média i všechny problémy, které obnáší, a efektivně ji využívat ke svému blahobytu.

5. Role obrazu v kultuře. Výuka pomocí obrazu.

OBRAZ A MALBA PROVÁZÍ ČLOVĚKA OD POČÁTKU DĚJIN. Nejstaršími obrazy, které známe v dnešní době, jsou malby v jeskyni Chauvet ve Francii (31 000 ± 1300 p. n. l.). Od této chvíle člověka neustále provází obraz, který v průběhu času plní různé informativní a estetické funkce. Důležité vzdělávací role vizualizace si můžeme všimnout ve středověku. Jako typický příklad tehdejšího „obrazového vzdělávání" může posloužit Biblia pauperum (Bible chudých). Důležité vzdělávací role vizualizace si můžeme všimnout ve středověku.

Ke spojování obrazu s textem za účelem komplexnějšího znázornění konkrétního příběhu docházelo velmi často. V průběhu století se na mnoha obrazech objevovaly symboly, alegorie a emblémy, které zastupovaly text, naopak sám text byl nejednou doprovázen obrazem, často ve formě iniciál nebo iluminací.

Ke spojování obrazu s textem za účelem komplexnějšího znázornění konkrétního příběhu docházelo velmi často. V průběhu století se na mnoha obrazech objevovaly symboly, alegorie a emblémy, které zastupovaly text, naopak sám text byl nejednou doprovázen obrazem, často ve formě iniciál nebo iluminací.

Do tradice výuky s pomocí obrazu se zapsal také Jan Ámos Komenský, který v roce 1658 napsal Orbis sensualium pictus. Jednalo se o ilustrovanou „encyklopedii", rozdělenou do 150 kapitol se 150 dřevoryty. Svět v obrazech ukazoval a pojmenovával všechny věci duchovního a materiálního světa, které byly dětem potřebné.

Vizualizace

Obrazy mohou plnit nejen roli ilustrace, ale někdy je také možné s jejich pomocí vysvětlit problémy, které je těžké si představit. Jako příklad takového přístupu může posloužit text prof. Janusze Rachonia z Gdaňské polytechniky, ve kterém popisuje, jakým způsobem vysvětluje studentům teorii rezonance Linuse Paulinga pomocí obrazů Salvadora Dalího.

6. Mediální komunikace. Sociální komunikace: verbální, nonverbální komunikace.

Mediální výchova

Vztah mediální výchovy a mediální gramotnosti je jednoduše řečeno vztahem prostředku a cíle. Mediální výchova je tedy definována pomocí mediální gramotnosti jako výchova k orientaci v masových médiích, k jejich využívání a zároveň k jejich kritickému hodnocení jako záměrné výchovné působení na dosažení určitého stupně mediální gramotnosti, nebo také jednoduše jako výchova k životu s médii. Mediální výchova se utvářela postupně a její kořeny jsou kladeny někdy ke Komenskému, jindy až do antického Řecka, nicméně skutečný rozvoj přišel až po druhé světové válce. Mediální komunikace má obrovskou moc, vytváří svou realitu; využívá se ke zvýšení rozsahu působení nebo k sebeprezentaci; je dobré udržovat příznivý vztah s médii; sousední obory – public realitons a media realitons.

Typy komunikačních médií

- primární – „soustavy znaků a pravidel pro jejich používání (přirozený jazyk)
- sekundární – prostředky snažící se o záznam a přenos sdělení (obrázky, písmo, tisk, přenosová a vysílací technika, počítačové komunikační sítě).

Slovo komunikace pochází z latinského communicare, jenž znamená „společně něco sdílet, činit něco společným. Komunikace je základní podmínkou existence každého sociálního vztahu. Je také prostředkem pro sociální začlenění jednotlivce do lidského společenstva. Pro dokonalou interakci s okolím je nezbytné naučit se naslouchat vnitřnímu impulsu – vědomě pozorovat svoje myšlenky, kultivovat neustále vnitřní rozhovor, monitorovat pocity a naučit se jim rozumět.

Komunikaci můžeme rozdělit na:

- *Intrapersonální komunikace* probíhá uvnitř jedince a má podobu vnitřního dialogu. Jedná se o rozmlouvání "sám se sebou", sebereflexi vlastního jednání a komunikování s vnějším okolím.
- *Interpersonální komunikace* probíhá mezi dvěma a více lidmi, mezi kterými existuje nějaký vztah. Specifickým druhem interpersonální komunikace je komunikace skupinová.
- *Masová komunikace* je charakteristická jednosměrným prouděním informací od jednoho a více komunikátorů (zdrojů) k mnoha komunikantům (příjemcům). Jde o komunikaci uskutečňovanou skrze média, jakými jsou například: rádio, televize, tisk a internet. Při masové komunikaci nedochází k přímé zpětné vazbě, je proto důležité přijaté informace kriticky zhodnotit.

Vybíral (2009) definuje pět základních funkcí komunikace: *informovat, instruovat, přesvědčit, vyjednat, pobavit.* Sociální komunikace je podmínkou a předpokladem existence jakéhokoliv lidského společenství.

Činitelé komunikace:

- zdroj komunikace - komunikátor,
- určení komunikace /příjemce/
- komunikant,
- sdělení /komuniké/,
- prostor komunikace - kanál.

7. Mediální výchova jako obrana proti negativním vlivům mediálních komunikací.

Vztah mediální výchovy a mediální gramotnosti je jednoduše řečeno vztahem prostředku a cíle. Mediální výchova je tedy definována pomocí mediální gramotnosti jako výchova k orientaci v masových médiích, k jejich využívání a zároveň k jejich kritickému hodnocení jako záměrné výchovné působení na dosažení určitého stupně mediální gramotnosti, nebo také jednoduše jako výchova k životu s médii. Mediální výchova se utvářela postupně a její kořeny jsou kladeny někdy ke Komenskému, jindy až do antického Řecka, nicméně skutečný rozvoj přišel až po druhé světové válce.

Důvody pro mediální výchovu

- mnoho informací, poznatků, názorů;
- měnící se podmínky lidského života;
- vyvinutí mediální gramotnosti jako součásti všeobecného vzdělání;

Mediální gramotnost = soubor kompetencí, které napomáhají uživateli vyhledávat, analyzovat a hodnotit informace a dále je předávat; obsah a způsob realizace jsou odlišné → systém vzdělávání.

Koncepční otázky:

- Co má být obsahem mediální výchovy?
- Jaké je poslání mediální výchovy?
- Jaký je způsob realizace mediální výchovy?

Koncepce mediální výchovy

dvě základní složky:

- *poznatkové* - typické pro Canadu a Skandinavii, vychází z kritické větve teorií médií, frankfurtské školy, britských kulturálních studií.
- *dovednostní* – typické pro USA, předpokládá, že si žáci osvojí potřebné poznatky a dovednosti tím, že si sami vyzkouší, jak média fungují.

Negativní vlivy mediálních komunikací

Rozhlas

Domnívám se, že role rozhlasu jakožto prostředku ovlivňování názorů a hodnotového systému dětí a mládeže je v současné době nejmenší ze všech informačních technologií. Většina mladých lidí používá rozhlas pouze jako zvukovou kulisu. Jediné, co může mít na posluchače vliv, jsou nezbytné reklamy, které se jim často velice podbízí vtíravou melodií a vryjí se do paměti posluchače, a některé slogany se jim neustále vybavují.

Televize

Jedním z nejdůležitějších médií působících na psychiku dítěte je stále televize. Když se dítě dívá na televizi 24 McLUHAN, Herbert na nějaký akční film se špatným obsahem, považuje všechno, co se v něm děje za samozřejmost a leccos z toho si chce také vyzkoušet, aniž by domyslelo důsledky.

DVD přehrávač

Děti tráví svůj čas díváním se na dvd disky místo toho, aby šly třeba ven s kamarády, ale tato záliba může vést i ke kopírování dvd disků, která je z hlediska autorských práv nelegální. Děti sledující film na dvd si u filmu nerozvíjí své rétorické dovednosti, nezapojují fantazii. Což může vést k malé slovní zásobě, špatnému vyjadřování a obezitě.

Mobilní telefon

U mobilního telefonu je nebezpečí vzhledem ke školnímu prostředí, že žáci, studenti budou svůj telefon zneužívat k podvádění. Vede k nepozornosti při hodině, k nesoustředění. Často slouží k hraní her, k psaní sms při výuce.

Příklad negativních vlivů

- každodenní komunikace se odehrává prostřednictvím elektronické pošty,
- finance spravujeme pomocí e-bankingu,
- zprávy čteme převážně na internetu,
- vliv médií na psychiku jednotlivce - především na děti a mládež,
- sociální sítě – děti,
- nejúčinnějším způsobem jak zmírnit negativní vlivy je rodina.

8. Fotografie, její historie. Digitální současnost a budoucnost.

Historie fotografie

Už staří Řekové zjistili, že když světlo prochází skrz malý otvor do temné místnosti, zobrazí se v ní obraz toho, co je venku. V 9. století př. n. l. tento vynález používali Arabové v astronomii při určování polohy Slunce nebo slunečných zatmění. Leonardo da Vinci, italský učenec 15. století, popsal podrobně tento jev jako "camera obscura", čili temná místnost.

V 16. století Ital Giambattista Della Porta použil namísto malého otvoru čočku, čímž získal ostřejší obraz. Od té doby se camera obscura vyskytovala ve všech velikostech a to od nejmenších kapesních, až po ty velké umístěné na rozhlednách a majácích.

16. - 17. století přišli chemici na to, že některé látky, pokud se ponechají v otevřeném prostoru, mění svou barvu.

1725 - Johann Heinrich Schulz objevil, že soli stříbra jsou citlivé na světlo. Trvalo ovšem téměř dalších sto let, než kombinací těchto dvou samostatných objevů vznikl první obraz na papíře.

S vynálezem fotografie vznikl nový druh obrazu, v němž podstatnou roli při jeho vzniku hraje technika a mechanizace přejímající manuální práci při tvorbě záznamu skutečnosti. Nová technologie nebyla už založena na rukodělném, nýbrž fotochemickém procesu.

Digitální fotografie: Vznik technologie digitálního fotoaparátu je spojen s technologií záznamu televizního obrazu. V roce 1951 poprvé páskový videorekordér (VTR) zaznamenal obraz z televizní kamery konverzí do elektrických impulsů a uložil je na magnetickou pásku v laboratořích Binga Crosbyho (výzkumný tým sponzorovaný Crosbym vedený Johnem Mullinem). V roce 1956 se zdokonalila technologie VTR díky objevu Charlese P. Ginsburga (Ampex Corp.) a dostala se do běžného užívání v televizním průmyslu. Televizní a video kamery využívají velmi podobnou technologii, jako CCD snímač u digitálních fotoaparátů.

V polovině 70. let objevila firma Kodak několik senzorů schopných snímat statické obrázky, které pracovaly také na principu převodu světla. V r. 1986 objevili vědci Kodaku celosvětově první megapixelový(MPix) senzor, schopný zaznamenat 1.4 milionů pixelů (což znamená záznam fotografie 10x15 cm ve fotografické kvalitě). V roce 1990 vyvinul Kodak systém Photo CD – první celosvětový standard pro definici barev v digitálním prostředí počítačů (a potažmo periferií – tiskáren). Především v letech 2000 a 2001 vyrostla digitální fotografie po technologické stránce. Od snímačů s rozlišením 0,3 MPix až ke komerčně nabízenému profi CMOS snímači s rozlišením 22 MPix. Od plastových objektivů ke špičkovým ultrazoom objektivům.

Důležité osobnosti ve fotografii

- Joseph Niecephore Niepce

- Luis Jacques Mandel Daguerre
- F.S. Archer

9. Film, jeho historie, vývoj a budoucnost. Filmová technika.

Film je souhrnné označení pro kinematografii a veškeré její aspekty - umělecké, technické, komerční a společenské.

Způsob předávání informací, zdroj zábavy, prostředek komunikace.

Při sledování filmu vnímá lidské oko sled po sobě jdoucích obrázků. Rychlé střídání promítaných obrázků za velmi krátký časový úsek se našemu očnímu nervu a následně i mozku jeví jako nepřerušená činnost (obraz se netrhá a je vnímám mozkem jako spojitý pohyb).

Historie

- 1893 – kinetoskop – William Kennedy Laurie Dickson.
- Prohlížečka pro jednoho diváka.
- Film bylo možné sledovat jen skrze malé kukátko na těle přístroje.

K rozvoji filmu, jako technickému vynálezu, jako druh umění, kulturního, společenského a hospodářského fenoménu došlo koncem 19. století. Dějiny filmu jako nezávislé umělecké formy začínají po roce 1895, kdy byl s velkým ohlasem v Paříži předveden kinematograf bratří Lumièrů / 28. prosince 1895 v Paříži /. Použit byl film šířky 35 mm a rychlost 16 snímků za sekundu (fps). Bratři Lumiérové se zabývali dokumentárními a cestopisnými filmy. Za první krátký dějový film je považován jejich Pokropený kropič. Komedie se zahradníkem a zavlažovací hadicí

Film vznikl jako technický zázrak, nejmladší, syntetický a audiovizuální druh umění, jehož hlavním atributem je *ikoničnost.* Vynálezcem filmu je T. A. Edison, ale prapočátky filmu se dají najít už u tzv. oživené fotografie. Existují dvě základní paradigmata: *pasivní zachycování reality* (dokumentaristická linie – Lumiére) a *sci-fi, triková linie* (Mélies). Georges Méliès – roku 1897 v Evropě první filmový ateliér. Autor filmových triků.

Ve 20. letech 20. století došlo ke změně, která zcela změnila podobu filmu. V Kalifornii nastala doba zakládání filmových studií. Kvalita filmů hrála v nastalém konkurenčním boji stále větší roli a bylo třeba získávat diváky něčím, co se lišilo od průměrné produkce.

Různé velikosti záběru a filmového střihu umožnilo stupňovat filmové napětí a vést vyprávění na několika rovinách, což bylo například v divadle těžce realizovatelné. Prohloubení a vytříbení filmového vyjadřování v oblasti střihu a využití kamery. Díky využití více objektivů se plně využila škála záběrů od velkého celku až po detail, kamera se taktéž odpoutala z místa a byla používána k jízdě. V tomto období se film plně odpoutal od divadla, našel svůj vizuálně rytmický způsob vyjadřování, ale také nový filmový způsob herectví, psaní scénářů apod.

Němý film

Charlie Chaplin - jeden z nejslavnějších a nejlépe placených umělců. Pokaždé hrál

sociálně deklasované charaktery, které se lstí prosazují proti krásným a úspěšným lidem.

Joseph "Buster" Keaton, vytvořil nový, zcela osobitý typ filmové komiky. Charakteristickým znakem jeho filmové postavy byl strnulý, kamenný výraz tváře, který se stal jeho „firemní značkou“ a za který si vysloužil přezdívku „The Great Stone Face“ (Kamenná tvář).

Zvukový film

Na začátku 90. let 19. století, Edison a Dickson navrhli prototyp zvukového filmu, systém nazvaný Kinetofonograf nebo Kinetofon – předchůdce kinetoskopu poskytující nesynchronní zvuk. První známý (a také jediný přeživší) film se živě zaznamenaným zvukem byl test kinetofonu - celkově 17. pokus Dicksona o zvukový film.

Film Jazzový zpěvák (1927), od firmy firmou Warner Bros., zahájil éru zvukového filmu. V následujících pěti letech se od základu změnila veškerá technika, přístrojové vybavení i tvář filmu. Zvuk vzbudil větší zájem diváků o film. Práci ovšem ztratili mnozí herci nemluvící anglicky, nebo také pianisté, kteří vytvářeli v němých kinech zvukový doprovod.

Sovětský zvukový film se projevil o něco později. Toto zpoždění způsobila tvorba vlastního kinematografického materiálu, díky němuž nemuseli Sověti platit poplatky za materiál z ciziny. Ještě roku 1930 byly natáčeny němé filmy, např. Kulička. První anglický zvukový film Její zpověď natočil Hitchcock roku 1929.

30. léta byla velká éra klasického Hollywoodu, kdy vznikaly četné nové žánry a postupně se stával "továrnou na sny", jejíž napínavé či veselé filmy dovolovaly lidem krátkodobý únik z deprimující reality jejich všedního dne. Téměř všechny žánry reprodukovaly mýtus o zemi neomezených možností a přísahaly v nekonečných variantách na "americký sen" velké kariéry "od umývače talířů k milionáři.

Filmová produkce v Československu

Do konce roku 1941 pak byly zlikvidovány všechny české výrobní firmy a zůstaly jen dvě produkční společnosti - Lucernafilm a Nationalfilm. Všechny půjčovny byly sjednoceny do monopolního podniku Kosmos, který tak měl vedle Lucernafilmu a Nationalfilmu jediný právo k distribuci českých filmů.

10. Auditivní média, jejich historie, současnost a budoucnost.

Auditivní pomůcky

Podle Janíkové (2005) představují auditivní pomůcky důležitou úlohu při „zprostředkování a nácviku fonologického aspektu lexikální jednotky.“ (Janíková 2005, s. 129) Také pomáhají žákům na začátku procesu osvojování cizího jazyka, a když se

seznamují s novou slovní zásobou. Při použití autentického materiálu si žák může pomocí auditivních pomůcek poslechnout výslovnost rodilých mluvčích. Jako předmět auditivního média lze použít nahrávky na magnetofonu nebo CD, zprávy nebo reklamu z rádia nebo televize a další. Pro mnoho žáků zprostředkovávají auditivní média poslech autentického záznamu např. rodilého mluvčího.

Prostředky auditivní techniky

prostředek DT	pomůcka (nosič informace)
gramofon	gramofonová deska
kotoučový magnetofon	magnetofonový pásek
kazetový magnetofon	magnetofonová kazeta
přehrávač CD	CD (kompaktní disk)
rozhlasový přijímač	rozhlasové vysílání
audioknihy	mp3 záznam

Prostředky audiovizuální techniky

prostředek DT	pomůcka (nosič informace)
filmový projektor	zvukový film
televizor	televizní vysílání
videomagnetofon	videokazeta
multimediální počítače	CD-Romy s multimediálními programy
dataprojektory	data v počítačích či jiných zdrojích
DVD přehrávač	DVD

Auditivní média

Auditivní = sluchový; Média = střední, mezi, uprostřed. Médium je prostředníkem, něco zprostředkovává.

Definice - je těžké vymezit pojem; existuje mnoho definic. Úloha médií: sdělení, transport informací. Termínem „médium“ označujeme zpravidla technické prostředky a systém sociálních institucí sloužících komunikaci. Další možnou definicí tohoto pojmu je, že „média jsou společenské instituce, které se ve velkém podílejí na zajišťování komunikace ve veřejné sféře, a tím přispívají k rozvoji, ustavování a proměnám kultury, tedy sdílených významů, hodnot a výkladů světa.“(JIRÁK, J., KÖPPLOVÁ, B., 2003. 208, s. 52)

Co to jsou audiovizuální média?

Tento termín má dva významy. Můžeme tím označovat nosiče informací s audiovizuálním obsahem – tzn. působícím na sluchové a zrakové orgány (například DVD, videokazety), nebo jde o média vysílající sdělení, které na tyto smysly působí. 4 Milan Šmíd uvádí, že jde o odborný termín v rámci dělení médií, který se používá ve Francii, kde mezi audiovizuální média patří i rozhlas, což může být trochu problematické, jelikož rádio přenáší pouze auditivní informace.

Na druhou stranu anglosaská literatura dělí média na tištěná a elektronická. Do kategorie elektronických médií patří rozhlas, televize, audiovizuální a auditivní záznamy. Americká literatura při tomto dělení zahrnuje mezi elektronická média i film, který měl v minulosti s elektronikou pouze málo společných rysů. V současnosti je však jiţ úzkými vazbami propojen s médii elektronickými.

Současnost

Auditivní média - stále využívána, často propojena s vizuálními → audiovizuální média.

Přehrávače - telefony, tablety, počítače, televize, mp3/4, přehrávače, rádia, autorádia, diktafony, člověk :-).

Audiokniha = zvukový záznam mluveného slova; vznikla již v historii, nyní boom díky chytrým telefonům, internetu → stahování.

Budoucnost

Očekáváme nějakou budoucnost auditivních medii? Budou součástí naší domácnosti TV obrazovky či rádia, jako je tomu dosud? Budou TV v našich domácnostech nahrazeny projektory?

Vzniká nový koncept: projektory které pro zobrazení budou potřebovat málo místa - obrovský obraz na vzdálenost 25cm, odpadá složitá instalace, připojení k set-top-boxu, počítači či herní konzole bezdrátovým systémem; projektory do mobilů a tabletů /Lenovo,.../. Snížení ceny projektorů umožní nahrazení televizorů.

11. Televizní svět v historii. Analog a digitál.

Historie televize v ČSR

Průkopníci:

František Pilát - pozdější poválečný technický ředitel filmového studia Barrandov, sám si sestrojil televizní přijímač. Pilát jako první v Československu přijímal experimentální Bairdovo "třicetiřádkové" vysílání, šířené počátkem třicátých let (1929-1935) z Velké Británie na střední vlně 261,5 metru.

Nejaktivnější předválečný průkopník televize je dr. *Jaroslav Šafránek*, docent experimentální fyziky na Univerzitě Karlově v Praze. V roce 1935 sestrojil Šafránek vlastní fungující televizní zařízení, s nímž později cestoval po republice a veřejně ho předváděl. Ministerstvo pošt a telegrafů odmítalo Šafránkovi dát povolení k vysílání televize do éteru. Šafránkovo zařízení mohlo pracovat jen v laboratoři a přednáškových sálech. Zatímco Šafránek, radioamatéři a jejich zájmová organizace Československý radiosvaz žádali povolení pro experimentální vysílání mechanické nízkořádkové (30 řádek) televize sloužící hlavně radioamatérům, ministerstvo pošt a telegrafů, které od roku 1934 bedlivě sledovalo vývoj v zahraničí, chtělo poskytnout frekvence k televiznímu vysílání až nějakým rozvinutějším projektům. Řídilo se zásadou - vyčkávat, studovat zahraniční skutečnosti a pak teprve rozhodnout.

V roce 1939 byly ukončeny televizní výzkumy na území bývalého Československa. (Ohrožení republiky, Mnichov a nacistická okupace). V té době Šafránek pracoval na dokonalejším zařízení s rozkladem obrazu na 240 řádků.

17. listopadu Němci uzavřeli české vysoké školy.

Šafránkovy experimenty s televizí skončily. Ztratil svoje místo na univerzitě a jeho laboratoř ve Fyzikálním ústavu německé úřady uzavřely. Šafránkovi se prý podařilo některá zařízení odvést do pardubické továrny Telegrafia, kde zůstala po celou dobu války.

Ještě před koncem války, v dubnu 1945 opouštějí závod Fernseh A. G. špičkoví němečtí odborníci. Přesunují se do Rakouska. V květnu obsazují závody české místní úřady, závod ve Smržovce Fernseh A. G. Se okamžitě přejmenuje na Televid. Začátkem června přebírá Televid pod svoji správu československé ministerstvo obrany.

V červenci 1945, se však ostrahy podniku ujímá sovětská vojenská správa, pro níž je závod součástí válečné kořisti, přichází sem i několik sovětských odborníků z Leningradského televizního institutu. V této době do Smržovky často zajíždí z Prahy i docent Šafránek. Podle vzpomínek pamětníků však do technického vývoje už nezasáhl, neboť jeho mechanický systém s 240 řádky byl již zastaralý a v Televidu se pracovalo na elektronické, pozdější "evropské" normě 625 řádek. Šafránkovo jméno se však ještě jednou zapsalo do dějin naší televize. Organizoval stáž skupiny 25 odborníků, kteří do Smržovky - po dohodě českých úřadů se sovětskou vojenskou správou - v říjnu 1945 nastoupili. Dříve než se stážisti mohli rozkoukat a do prací aktivně zasáhnout, sovětská

strana se rozhodla Televid jako válečnou kořist odstěhovat.

Jaroslavu Šafránkovi se připisuje prvenství ve věci popularizace televize v Československu. Vydal knihu Televise, v níž seznamoval s technickými principy přenosu obrazu na dálku. Aktualizovaná verze s názvem "Televise - fysikální a technické základy televise" Šafránkovi vyšla po válce. Šafránek se pokusil odlišit prostou technologii přenosu pohyblivého obrazu od komplexního procesu televizního vysílání, když pro televizi jako masové médium razil slovo "rozjev", které podle něj "správně vystihuje podstatu televize.

23. března byli do Tanvaldu sezváni 1948 novináři, které zde uvítal za VTÚ generál Josef Trejbal a za Český rozhlas technický náměstek Kazimír Stahl. Součástí předvedené techniky vlastní konstrukce, byť s využitím některých trofejních součástek, byl i televizní přijímač s obrazovkou vlastní výroby o rozměrech 16x21 cm.

Na výstavě MEVRO v Praze pracovaly dvě televizní kamery a signál se přenášel k přijímačům ještě kabelem. Měsíc po skončení výstavy MEVRO, 4. července 1948, při přenosech z XI. Všesokolského sletu pracovaly na Strahovském stadionu již tři kamery a signál se vysílal vzduchem ze stožáru na Petříně pro 25 přijímačů v různých institucích a na veřejných místech (např. Výstaviště, budova Čs. rozhlasu, redakce Rudého práva, příjem se kontroloval též mimo Prahu - v jižních Čechách a v Krkonoších).

Od roku 1949 do roku 1952 jakoby televize v Československu přestala existovat.

Televizní zařízení Vojenského technického ústavu VTÚ včetně dvou kamerových řetězů a deseti televizních přijímačů se převedlo do Československého rozhlasu, který začátkem roku 1949 zřizuje Ústav rozhlasové techniky ÚRT. Ten sice i nadále plní úkol připravovat televizní vysílání v průběhu první pětiletky, tj. do konce roku 1953, ale studená válka, která se vyostřila v roce 1950 konfliktem v Koreji, způsobila, že ÚRT neměl s kým spolupracovat, protože na základě pokynu Ministerstva obrany se technický výzkum orientoval výhradně na vojenské potřeby. Podle Ing. Františka Křížka ÚRT v roce 1951 organizoval ve své budově ve Vokovicích několik pokusných vysílání a zapůjčil přijímače stranickým a vládním činitelům, aby televizi propagoval, ale bez úspěchu. K obratu dochází v roce 1952.

Vláda 8. 4.1 952 vydává nařízení, kterým uložila ministerstvu spojů péči "o výstavbu a provoz technických rozhlasových a televizních zařízení“.

První "programový a provozní ředitel televizního studia" zřízeného v rámci Československého rozhlasu Karel Kohout byl jmenován1.února 1953 (tři měsíce před plánovaným začátkem vysílání). Karel Kohout přišel z barrandovského studia a začal se svoji legendární sekretářkou Marii Kořenovou organizovat vysílání z provizorní kanceláře na Václavském náměstí.

Vysílání se několik let omezovalo na petřínský vysílač v Praze s dosahem do středočeského kraje až k podhůří Jizerských hor a Krkonoš. Na konci roku 1953 bylo v provozu asi 2000 televizorů, z nichž tisícovka značky Leningrad se dovezla z NDR, kde se tehdy v sovětské licenci vyráběly. Ale už v roce 1953 dodávala strašnická Tesla na trh přijímače Tesla 4001A. Prodávaly se za 4000 Kč (tehdy představovalo téměř půlroční

průměrný plat). Začátkem ledna roku 1955, když se začal platit tzv. koncesionářský poplatek, uvádí statistika 3833 přihlášených koncesionářů. Koncem padesátých let se televizní přijímač stal nedostatkovým zbožím na trhu.

11. února 1955 televizní probíhá první přímý televizní přenos sportovního utkání v historii československé televize.

17. dubna 1955 se uskutečnil první přímý přenos opery z Národního divadla.

Od října 1955 se vysílalo 6x týdně (nevysílalo se v pondělí); od 29. prosince 1958 se začalo vysílat celostátně každý den, sedm dnů v týdnu.

O Silvestra 31. prosince 1955 zahájil provoz druhý československý televizní vysílač Ostrava-Hošťálkovice.

Televizní studio v Brně se zřídilo až v roce 1961; 25. února 1962, začalo televizní vysílání v Košicích.

Tato základní struktura pěti hlavních televizních studií vydržela v podstatě až do rozpadu Československa v roce 1993. Na přelomu 50. a 60. let se bouřlivým tempem budovala síť vysílačů a převaděčů tak, že už v roce 1961 televizní signál pokrýval všechna regionální centra a většinu Československa, takže krátce nato (1962) byla překročena hranice jednoho miliónu majitelů televizních přijímačů.

1. ledna 1993, po rozpadu federace, vznikla Česká televize. V 90. letech zahájily vysílání první soukromé TV společnosti (NOVA a další). V roce 2000 se Česká republika připravovala na přechod k digitálnímu TV vysílání v digitální zemské (dříve se používal termín pozemní) *platformě DVB-T*, která měla nahradit analogové zemské vysílání. Technicky byla Česká republika připravena velmi dobře – ve třech největších městech probíhalo již úspěšně experimentální digitální televizní vysílání.

Rozdíl mezi analogovým a digitálním přenosem je možné přirovnat k posílání peněz dostavníkem, anebo bankovním převodem. Pokud pošleme peníze dostavníkem, obdrží adresát naše peníze fyzicky tak, jak jsme je zaslali, ovšem za předpokladu, že ten dostavník někdo nevykrade, že je kurýr někde neztratí apod. Zatímco, když je pošleme bankovním převodem, obdrží adresát sice fyzicky jiné peníze, ale v každém případě ve stejné hodnotě, v jaké byly odeslány.

Digitální přenos informací je přenos údaje o hodnotě, a to v podobě čísla. (v tomto případě tzv. binárního čísla).

Analogové vysílání je dnes již zastaralý způsob šíření televizního a rozhlasového signálu. Obraz i zvuk je přenášen pomocí elektromagnetického vlnění. Informace o barvách a zvuku jsou vytvářeny modulací tohoto spojitého (analogového) signálu. Každá televizní nebo rozhlasová frekvence tak nese signál jedné stanice. Analogové vysílání je v současnosti nahrazováno digitálním vysíláním. V České republice bylo analogové vysílání vypnuto ke konci roku 2011, kdy nastala tzv. analogová tma.

Digitální vysílání (DVB = Digital Video Broadcasting) umožňuje pomocí multiplexu přenášet na jedné frekvenci několik televizních programů. Umožňuje tím lépe využít přenosové pásmo používané při analogovém televizním vysílání.

Multimédia v zrcadle doby. Filozofie médií.

Co jsou média? Média bývají označována jako prostředek komunikace, sdělovací prostředek, nejčastěji technické zařízení umožňující komunikaci mezi komunikátorem a recipientem. A dále jsou to především hromadné sdělovací prostředky, tzv. nositelé propagace. V propagaci hrají důležitou úlohu i jiná média než masová. A jako příklady můžeme uvést veletrhy, výstavy, výkladní skříně, obaly, ale i přednášky, exkurze.

Denní tisk, časopisy, filmy, rozhlasové a televizní vysílání, internet (zpravodajské, výukové, zábavné portály, sociální síť, blogy, blogy, mluvené slovo, filmy, hudba) – komunikační prostředky, které jsou k dispozici potenciálně velkému počtu uživatelů, a to v pravidelných intervalech.

Pomocí nich můžeme cíleně ovlivňovat veřejné mínění, může docházet k manipulaci, k záměrnému výběru zpráv a jejich zkracování, ke směšování zpráv a hodnotících komentářů, k přehánění (kladná zpráva vyzní kladněji, negativní záporněji).

Síla střihu (ze souvětí je ponechána jen jeho část), manipulace s obrazem (zpřeházením pořadí obrazů může dojít k záměně příčiny a následku). Zkreslování slovního doprovodu.

Funkce médií

- informační funkce. Plní ji především zpravodajství a publicistika, kde se využívají zprávy, komentáře a reportáže.
- zábavná funkce. Masmédia přinášejí hudbu, umění, sportovní přenosy, ale i fenomény dnešní doby – televizní seriály, reality show, vaření.
- komerční funkce zajišťuje médiím finance na vysílání a zaručuje zisk, realizuje se především reklamou, ale také teleshoppingem a sponzoringem.
- výchova, edukace.

Politické, sociální působení.

1900 - 1925 vzniklo novinářské odvětví, první reportáže z terénu a fotoreportáže.

1918 - založena Československá tisková kancelář „ČTK“; politické deníky, nezávislé deníky, bulvár. Přestože je dnes systém první republiky pro mnohé vzorem demokratického zřízení, fungování periodického tisku bylo v tomto období omezováno daleko silněji, než je tomu například dnes.

Role rádia – zábava, zpravodajství

V roce 1925 začalo zpravodajství hrát v rozhlase novou a aktivnější úlohu, už to nebylo jen hlášení zpráv, ale snaha přiblížit události přímo. První pokusy o reportáž. První sportovní reportáž.

Rozhlas a vzdělání

/Postupně se do programové nabídky zařadily také rozhlasové kurzy cizích jazyků, např. francouzštiny/. Přednášky, rozhovory, besedy, pásma, rozhlasové hry, odborné vysílání.

1930 – 1960

Televize

Průkopníkem u nás byl ve 30. letech Jaroslav Šafránek (docent experimentální fyziky na Univerzitě Karlově v Praze). V roce 1935 dokončil první přijímací televizní aparaturu v Československu. Velký zájem o televizi projevovali českoslovenští radioamatéři.

Síla rozhlasu pokračuje

1939 – podřízení rozhlasu protektorátním úřadům (musejí odejít všichni zaměstnanci židovského původu).

Film a válečná popaganda

Vznikaly série filmů líčících teritoriální expanzi Třetí Říše. Tyto filmy neměly žádnou velkou uměleckou hodnotu.

Média a komunismus

V roce 1948 je ČTK pod komunistickou diktaturou. Slouží jako nástroj politické propagandy vládnoucí strany. Existuje silná cenzura, zestátnění rozhlasu.

Dva typy zpravodajství: pro veřejnost a neveřejné (pro vysoké stranické a státní funkcionáře). Agentura je formálně podřízena vládě, fakticky ji ale řídí ústřední výbor Komunistické strany Československa.

Léta 50. – zostřená ideologická válka

Na příštích 40 let začala média v Československu sloužit "lidu a komunistické straně“.

1952 – Vznik Hlavní správy tiskového dozoru.

1952 - Československý rozhlas začal rušit vysílání Svobodné Evropy (vysílání zahájeno 1950)

Média před okupací 1968

Rozhlas: přirozený civilní projev, kritika, otevřenost, příjem světových rozhlasových stanic – rozšíření nabídky, ale oslabení možnosti kontroly ze strany státu.

Televize: pouze jeden program, obrovský nárůst koncesionářů, „Cukrák“ – dokončení

výstavby základní sítě vysílačů.

1968 okupace - doba normalizace

ROZHLAS: rušičky zahraničních stanic (Svobodná Evropa, Hlas Ameriky).

TELEVIZE: vysílač pro druhý program, 1973 – první barevné vysílání, personální čistky, obnovení cenzury, ideologická koncepce, útoky na představitele disentu.

TISK: ilegální vydávání novin a časopisů.

Léta 80. a další

1986 – Černobyl.

1989 – Sametová revoluce.

Velkou změnou je vývoj osobních počítačů →

1981 – počítače 4. generace.

1990 – 2017

Počátek 90. let – transformace médií, konec cenzury - média směřují do osobního vlastnictví, média jsou institucí svobody projevu, fórem sloužícím k diskusím o věcech veřejného zájmu a oblastí soukromého podnikání. 1991 - představen 1. internetový prohlížeč – WorldWideWeb v CERNu.

ÚVOD DO PROGRAMOVÁNÍ JAVA

1. Úvod do programování v JAVA

Tvorba programů pro javovskou platformu probíhá ve dvou fázích. V prvním kroku napíšeme tzv. zdrojový kód. Zdrojový kód je zápisem programu. Ve druhém kroku je zdrojový kód přeložen do bajtkódu. Překlad zdrojového kódu do bajtkódu zajistí překladač (kompilátor, angl. compiler).

Bajtkód (angl. bytecode) jsou instrukce pro javovský virtuální stroj (angl. Java Virtual Machine, JVM).

1.1. Třídy

Program v Javě se skládá z jedné nebo více tříd. Třídu deklarujeme pomocí klíčového slova class. Třída může obsahovat metody. Každá třída i metoda má jméno (s výjimkou anonymních tříd, o kterých zde ovšem nebude řeč). Obsah třídy i metody je uzavřen ve složených závorkách { a }. Do metod zapisujeme příkazy. Každý program musí obsahovat metodu main, která má vždy stejnou hlavičku:

```
public static void main( String[] args )
```

Význam jednotlivých slov v této deklaraci probereme později. Pro začátek budeme zapisovat příkazy pouze do metody main. Tato metoda slouží jako vstupní bod do aplikace. Spuštění programu probíhá tak, že JVM zavolá tuto metodu. Příkazy v ní se provádějí jeden po druhém v pořadí, v němž jsou zapsány.

K výstupu na obrazovku slouží metoda System.out.println(). Program, který vytiskne jednoduchý pozdrav, vypadá takto:

```
public class Prvni {
   public static void main( String[] args ) {
      System.out.println( ”ahoj” );
   }
}
```

Tento zdrojový text deklaruje třídu Prvni, která obsahuje metodu main.

1.2. Javovský virtuální stroj

Javovský virtuální stroj (angl. Java Virtual Machine, JVM). JVM umožňuje spouštět javovské programy. Úkolem JVM je poskytovat programům vždy stejné prostředí. Stejné prostředí zajistí, že javovský program běží na počítači s procesorem SPARC a operačním systémem Solaris stejně jako na počítači s procesorem Intel a operačním systémem MS Windows.

Spuštění javovského programu probíhá ve dvou krocích: v prvním kroku spustíme JVM a ve druhém kroku JVM nahraje do paměti bajtkód a spustí jej (zavolá metodu main). Paměť programu je rozdělena na tři oblasti: zásobník (angl. stack), halda (angl. heap) a oblast metod (angl. method area). Oblast metod obsahuje program (bajtkód). Program je tvořen posloupností instrukcí JVM. Každá instrukce se skládá z jednobajtového operačního znaku a případně operandů. Např. instrukce goto má operační znak 167 a provede nepodmíněný skok na adresu určenou operandem.

Zásobník slouží pro alokaci lokálních proměnných, pro ukládání parametrů a výsledků instrukcí JVM a pro předávání parametrů a výsledků při volání metod. Paměť zásobníku se přiděluje po tzv. rámcích (angl. stack frames). Každé volání metody alokuje nový rámec. Rámec obsahuje paměť pro lokální proměnné a tzv. zásobník operandů (angl. operand stack). JVM udržuje ukazatel na tzv. vrchol zásobníku, což je naposledy vložená hodnota. Zásobník operandů se používá pro parametry a výsledky instrukcí JVM. Paměť běžícího programu Např. instrukce iadd vyzvedne dvě hodnoty z vrcholu zásobníku, tyto hodnoty sečte a výsledek umístí na vrchol zásobníku.

Každá položka operandového zásobníku má délku nejméně 4 bajty. Všechny aritmetické instrukce JVM tedy pracují s operandy délky nejméně 32 bitů. V oblasti proměnných se paměť přiděluje také po 4 bajtech. To znamená, že každá proměnná zabírá v JVM nejméně 32 bitů.

Halda je oblast paměti, v níž můžeme za běhu programu vytvářet objekty. Vytváření objektů na haldě se provádí pomocí klíčového slova new. Na rozdíl např. od jazyka C++ není potřeba objekty odstraňovat (tzv. dealokovat). O odstraňování objektů se v Javě stará tzv. garbage collector. Garbage collector sám rozpozná nepoužívané objekty a tyto objekty dealokuje. Po dealokaci objektu je paměť, kterou objekt zabíral, připravena k dalšímu použití.

Popis programovací jazyka rozdělujeme na dvě části: syntaxi a sémantiku. Syntaxe (angl. syntax) popisuje pravidla pro správný zápis. Např. říká, kde je nutné psát závorky. Je-li program syntakticky správně, lze jej přeložit a spustit. Sémantika (angl. semantics) definuje význam jednotlivých jazykových konstrukcí. Např. říká, že operátor = provede přiřazení. Aby program dělal to, co po něm chceme, musí být sémanticky správně. V tomto textu se budeme zabývat syntaxí i sémantikou. Probereme postupně většinu syntaktických konstrukcí jazyka Java a u každé si řekneme její sémantiku. Při zápisu programu používáme tzv. klíčová slova (angl. keywords). Jsou to slova, která mají v jazyce speciální význam (dvě z nich, const a goto, zůstávají ovšem nepoužitá). Java rozlišuje malá a velká písmena. Např. Class je něco jiného než class. Klíčová slova jsou pouze z malých písmen.

1.3. BlueJ

BlueJ je volně šiřitelné multiplatformní vývojové prostředí vyvinuté speciálně pro výuku objektově orientovaného programování v jazyce Java. Umožňuje studentům navrhovat diagram tříd vyvíjené aplikace ve zjednodušené verzi jazyka UML. Hlavní výhodou BlueJ je jeho interaktivnost – umožňuje vytvářet instance jednotlivých tříd, zasílat jim zprávy a volat jejich metody. (Wikipedia)

1.4. Instalace

Nedříve je třeba nainstalovat Javu. Z oracle.com si stáhněte Balík Java JDK (Java Development Kit.)

BlueJ je možné stáhnout na adrese BlueJ.org. Obojí si nainstalujte. Návod na používání programu BlueJ naleznete například na této adrese: https://www.bluej.org/tutorial/tutorial-czech.pdf

1.5. Tvorba proměnných

Do proměnných se ukládají hodnoty. Proměnná je pojmenované místo v paměti. V rámci tvorby proměnné – její deklarace, v programovacím jazyce Java musí být u každé proměnné nejdříve řečeno, jaký datový typ bude obsahovat viz příklad 1 v našem případě slova int, boolean, char. Jejich význam si vysvětlíme v dalším odstavci.

Následuje název proměnné. Názvy proměnných píšeme bez diakritiky a mezer. Mezery mezi slovy se prostě vynechávají a každé další slovo začíná velkým písmenem. Výraz uzavíráme středníkem.

Příklad 1

```
int a;
boolean IsTrue;
char CarConsumption;
```

V příkladu 1 jsme vytvořili proměnné, ale nepřiřadili jsme jim žádné hodnoty. V příkladu 2 jsme vytvořili proměnné, kterým jsme přiřadili konkrétní hodnoty. První přiřazení do proměnné nazýváme inicializace.

Příklad 2

```
int a=2;
boolean IsTrue =true;
char CarConsumption ='A';
```

Chceme-li deklarovat více proměnných najednou, musíme je oddělit čárkou:

```
int a, b;
```

Budeme rozlišovat mezi výrazem a příkazem. Výraz má vždy nějakou hodnotu. Tuto hodnotu získáme vyhodnocením výrazu. Např. 42 a x + 1 jsou výrazy. Příkaz je kód, který něco provede. Např. x = 1; je příkaz, který přiřadí hodnotu 1 do proměnné x. Máme-li výraz, při jehož vyhodnocení se něco provede, můžeme z něj obvykle udělat příkaz tak, že za něj napíšeme středník. V takovém případě říkáme, že vyhodnocení tohoto výrazu má vedlejší efekt.

2. Datové typy

2.1. Celo číselné

Int – Z anglického *integer* – celé číslo. Můžeme tedy zadávat pouze celá čísla v rozsahu od -2,147,483,648 do 2,147,483,647, včetně nuly. Integer může reprezentovat například počet operací počítače za sekundu.

Další celočíselné datové typy jsou vypsány v tabulce:

Typ	Počet bajtů	Rozsah	
byte	1	-128	127
short	2	-32 768	32 767
int	4	-2147 483 648	2147 483 647
long	8	-9,22 x 1018	9,22 x 1018

Stane-li se, že výsledek operace neleží v přípustném intervalu daného datového typu, nastává přetečení (angl. overflow). V takovém případě má výsledek opačné znaménko než výsledek matematické operace. V Javě nezpůsobí přetečení chybu, je však třeba s ním počítat. Např. pokud máme v proměnné typu byte hodnotu 127 a přičteme k ní 1, bude v proměnné −128.

2.2. Reálná čísla

Pro práci s reálnými čísly má Java dva primitivní datové typy: float a double. Oba používají stejný způsob reprezentace: reálné číslo je uloženo jako trojice znaménko, mantisa a exponent. Čísla tak ukládáme pouze přibližně. Typ float používá 32 bitů: 1 bit zabírá znaménko, 8 bitů exponent a 23 bitů mantisa. Typ double používá 64 bitů: 1 bit pro znaménko, 11 bitů pro exponent a 52 bitů pro mantisu.

2.3. Konverze

Konverze (přetypování) je převod hodnoty na hodnotu jiného datového typu. Např. převod hodnoty typu double na hodnotu typu int. Některé konverze se provádějí automaticky, např. z typu int na typu double, jiné si musíme vyžádat, např. z typu long na typ int. Automatické konverze není nutné zapisovat:

```
int a = 100;
long a = b;  // Automatic conversion from int to long
```

Je-li třeba konverzi předepsat, uvádíme cílový typ v závorkách před konvertovanou hodnotou. Např. konverzi z double na int zapíšeme takto:

```
double d1 = 5.85;
int i1 = (int) d1;
```

Výsledkem této konverze bude hodnota, která je v zápise původního čísla před desetinnou tečkou (pro nezáporné hodnoty je to celá část čísla). Tj. v proměnné i2 bude hodnota 5.

```
double d2 = -4.99;
int i2 = (int) d2;
```

V proměnné i2 bude hodnota −4. Připomeňme, že to není celá část čísla, protože celá část čísla −4.99 je −5.

2.4. Logické

Boolean - Obsahuje pouze výrazy true a false (Pravda a nepravda). Je také nazývána jako logická proměnná. Pokud bychom porovnávali dvě číselné proměnné ve výrazu a<b. Nazvěme je *a* a *b*. pokud je *a* skutečně menší než *b* pak je výraz a<b pravdivý a do proměnné c se uloží hodnota true. Pokud výraz pravdivý není do c se uloží false

```
int a = 5;
int b = 6;
boolean c = a<b;
System.out.println(c);
```

- V tomto příkladu systém vypíše „true“.
- Hodnota false je vnitřně reprezentována jako 0, hodnota true jako 1.

2.5. Textové

Char - Z anglického character – znak či symbol. Můžeme do něj ukládat jeden znak. Před tímto a za tímto znakem musí být apostrofy (jednoduché uvozovky). Viz příklad 2. Vypisovat můžeme znaky z tabulky Unicode. Hodnotu typu char lze chápat jako index (pořadí) znaku v tabulce Unicode.

```
char c ='H';
System.out.println(c) ;
```

2.6. Textové řetězce

Pro práci s řetězci používáme v Javě typ String. Řetězcové konstanty zapisujeme do uvozovek.

```
String s = " Hi how are you?";
```

Řetězce můžeme spojovat pomocí operátoru +.

```
String s1 = "jdk", s2 = "7.0";
String s3 = s1 + s2;  // Creates a string "jdk7.0"
```

K řetězci lze přičíst i hodnotu jiného typu. V takovém případě se hodnota nejprve převede na řetězec a poté se oba řetězce spojí.

```
int x = 42;
String s = "answer is " + x;
```

Příklad vytvoří řetězec "odpověď je 42".

2.7. Komentáře

V Javě jsou tři druhy komentářů:

- jednořádkový - začíná znaky // a pokračuje do konce řádky
- víceřádkový - začíná znaky /* a končí znaky */
- dokumentační – začíná znaky /** a končí znaky */

Dokumentační komentáře jsou určeny pro zpracování programem javadoc, který generuje dokumentaci ve formátu HTML.

```
/**
* Main program class.
*/

public class Main {
      public static void main(String[] args) {
            /* Prints the answer */
            System.out.println(42); // answer is 42
      }
}
```

Prostřednictvím komentářů vkládáme do zdrojového textu doplňující informace. Překladač komentáře přeskakuje, tudíž na běh programu nemají žádný vliv. Komentovat bychom měli např. význam proměnných, neobvyklé postupy a netradiční algoritmy. Tedy místa, jejichž význam nemusí být pro člověka čtoucího kód na první pohled zřejmý.

2.8. Terminálový vstup a výstup

V této kapitole si ukážeme, jak lze v Javě zapisovat na obrazovku a číst z klávesnice. K výstupu na obrazovku slouží tzv. výstupní proud (angl. output stream) System.out. Používat budeme tři jeho metody: print(), println() a printf(). Volání níže vypíše: Ahoj Babi

```
System.out.println( ” Hi Baby!” );
```

Metody print() a println() vytisknou hodnotu, kterou uvedeme v závorkách za jménem metody (této hodnotě říkáme parametr). Liší se tím, že println() k výstupu navíc připojí přechod na nový řádek a tudíž následující volání metody print() nebo println() bude tisknout od začátku řádky. Při tisku lze využít spojování řetězců pomocí operátoru +.

```
int v = 200;
System.out.println( ”Car moved ” + v + ” km/h” );
```

Elegantnější výstup nabízí metoda printf() (tzv. formátovaný výstup), která je obdobou stejnojmenné funkce jazyka C.

```
int m = 6;
System.out.printf( ” African elephant weighs %d tons”, m );
```

Parametry metody printf() jsou formátovací řetězec a seznam hodnot. Formátovací řetězec může obsahovat tzv. výstupní konverze, které určují, v jakém tvaru se vytiskne příslušná hodnota. Každá konverze začíná znakem %. Např. %d znamená výstup celého čísla v desítkové soustavě. Při provádění příkazu se místo konverze dosadí příslušná hodnota ze seznamu hodnot. Pokud hodnota chybí nebo neodpovídá výstupní konverzi, nastane chyba.

Zvláštním případem je konverze %n, které neodpovídá žádná hodnota v seznamu hodnot. Na výstupu se tato konverze projeví přechodem na nový řádek. V MS Windows se pro přechod na nový řádek používá dvojice znaků \r (carriage return) a \n (line feed). Unixové systémy používají znak \n. Konverze %n zajistí, že se použije správný přechod na nový řádek, tj. \r\n na MS Windows a \n na Unixu.

Pro reálná čísla máme konverzi %f. Můžeme u ní stanovit počet číslic za desetinnou tečkou (implicitní hodnota je 6). Např. %.2f vytiskne dvě číslice za desetinnou tečkou.

```
System.out.printf( ” Euler’s constant is about %.2f%n”,
Math.E );
```

Znaky tiskneme pomocí konverze %c.

```
char c = '@';
int i = c;
System.out.printf( "Character '%c' Is in the Unicode table
in position %d%n", c, i );
```

Pro řetězce používáme konverzi %s.

```
String Java = "Java";
System.out.printf( " Our favorite programming language
is %s%n", Java );
```

Pro čtení z klávesnice máme v Javě tzv. vstupní proud (angl. input stream) System.in. Většinou jej nepoužíváme přímo, ale např. přes třídu Scanner. Tato třída nabízí metody pro čtení primitivních typů a řetězců. Používáme-li třídu Scanner, náš program obvykle začíná příkazem import, kterým překladači říkáme, kde má třídu Scanner hledat. Před prvním čtením vytvoříme instanci třídy Scanner pomocí klíčového slova new. Načtení hodnoty typu int provedeme voláním metody nextInt() na této instanci.

```
import Java.util.Scanner;
public class Read {
      public static void main( String[] args ) {
            Scanner sc = new Scanner( System.in );
            int x = sc.nextInt();
            System.out.printf( " Readed value is: %d%n", x );
      }
}
```

Načtení hodnoty typu double zajistí metoda nextDouble(). Řetězec načteme metodou next().

```
Scanner sc = new Scanner( System.in );
double d = sc.nextDouble();
System.out.printf( ”Readed number is: %f%n”, d );
String s = sc.next();
System.out.printf( ”Readed string is: %s%n”, s );
```

3. Operátory

S čísly lze v Javě provádět běžné aritmetické operace: sčítání, odčítání, násobení, dělení, modulo (zbytek po celočíselném dělení). K zápisu operací slouží **operátory**. Např. sčítání zapisujeme pomocí operátoru + a modulo pomocí operátoru %. Zápis x + 2 je tedy zápisem operace sčítání. Každý operátor pracuje s jednou či více hodnotami nebo proměnnými, kterým říkáme **operandy**. Podle počtu operandů můžeme operátory rozdělit na unární (s jedním operandem), binární (se dvěma operandy) a ternární (se třemi operandy). Unárním operátorem je např. ++ (inkrementace) a binárním operátorem je např. = (přiřazení). Jediným ternárním operátorem je ?: (podmíněný operátor). Výsledkem provedení operátoru je hodnota (říkáme, že operátor vrací hodnotu). Např. binární operátor + (sčítání) vrací součet svých operandů. Typ výsledku závisí na operátoru a někdy i na operandech. U operátorů, které vrací celočíselnou hodnotu, je výsledek buď int nebo long. Např. sečteme-li dvě hodnoty typu int, výsledek bude int, sečteme-li dvě hodnoty typu long, výsledek bude long, a sečteme-li dvě hodnoty typu byte, bude výsledek int. Operátor / (děleno) vrací pro celočíselné operandy celočíselný podíl. Např. 15 / 4 je 3 a −15 / 4 je −3.

Je-li hodnota druhého operandu 0, nastane chyba. Je-li alespoň jeden operand reálný (tj. float nebo double), je i druhý operand převeden na reálný, a operátor děleno vrátí podíl obou operandů. Např. 4.5 / 3 je 1.5. Je-li druhým operandem 0, bude výsledkem reálného dělení některá z těchto hodnot: plus nekonečno, je-li první operand kladný, mínus nekonečno, je-li druhý operand záporný, nebo NaN (Not a Number), je-li první operand kladné nekonečno, záporné nekonečno, NaN či 0. Operátor % (modulo) vrací zbytek po celočíselném dělení. Např. 17 % 4 je 1. Tento operátor je definován i pro záporné hodnoty, ovšem jinak než tomu bývá v matematice. Znaménko výsledku je vždy stejné jako znaménko prvního operandu: −17 % 4 je −1, 17 % −4 je 1 a −17 % −4 je −1.

Operátory lze řetězit, tj. lze zapisovat výrazy, které obsahují více operátorů. Např. x + 2 * y je výraz, který obsahuje operátory sčítaní a násobení. Pořadí vyhodnocování operátorů určuje **priorita** operátorů (angl. precedence). Např. ve výrazu x + 2 * y se provede nejprve násobení a pak sčítání, protože operátor násobení má vyšší prioritu než operátor sčítání. Jiné pořadí vyhodnocení lze předepsat závorkami: (x + 2) * y. Použijeme-li ve výrazu více operátorů se stejnou prioritou, rozhoduje o pořadí vyhodnocování **asociativita** operátoru (angl. asociativity). Např. ve výrazu x − y − z se provede nejprve x − y a poté se od výsledku odečte z. Říkáme, že operátor odčítání asociuje zleva doprava. Výraz x − y − z má tedy stejnou hodnotu jako výraz (x − y) − z.

Některé operátory mají asociativitu opačnou, tj. zprava doleva. Příkladem je operátor přiřazení. Návratovou hodnotou tohoto operátoru je hodnota levého operandu po přiřazení. Ve výrazu x = y = 1 se provede nejprve přiřazení y = 1 a poté se návratová hodnota operátoru (v tomto případě 1) přiřadí do x. Výraz x = y = 1 se tedy vyhodnocuje stejně jako x = (y = 1).

3.1. Operátory inkrementace a dekrementace

Operátor inkrementace (++) způsobí zvětšení hodnoty proměnné o jedničku. Lze jej zapisovat dvěma způsoby: prefixově a postfixově. V prefixové notaci operátor předchází svůj operand, v postfixové jej následuje. V obou případech dojde k inkrementaci proměnné, rozdíl je však v návratové hodnotě operátoru. Prefixový operátor vrací hodnotu proměnné po zvětšení, postfixový operátor hodnotu před zvětšením.

```
int x = 1;
int y = ++x;
```

Ve výrazu y = ++x se provede nejprve operátor inkrementace, protože má vyšší prioritu než operátor přiřazení. Dojde tedy ke zvýšení hodnoty x o jedničku. Návratovou hodnotou tohoto operátoru je hodnota x po zvětšení, tj. 2. Ta se použije jako operand operátoru přiřazení. Do proměnné y se tedy uloží hodnota 2.

```
int x = 1;
int y = x++;
```

Ve výrazu y = x++ se provede nejprve operátor inkrementace. Jeho návratovou hodnotou je hodnota proměnné x před zvětšením, tj. 1. Do proměnné y se tedy uloží hodnota 1.

Operátor dekrementace (−−) způsobí snížení hodnoty proměnné o jedničku. Používá se obdobně jako operátor inkrementace.

```
int x = 1;
System.out.println( --x );
```

3.2. Logické operátory

Logické výrazy můžeme spojovat dohromady pomocí logických operátorů. Logický operátor má operandy typu boolean a vrací hodnotu typu boolean. Existují 2 logické operátory .

Operátor **logického součinu**, nazývaný též *a zároveň* anglicky *and* se zapisuje && a vrací true tehdy a jen tehdy, pokud mají oba jeho operandy hodnotu true.

Příklad:

```
if( x == 0 && y == 0 ) {
      system.out.println( ”x and y are equal to zero” );
}
```

Operátor logického součtu , nazývaný též *a nebo,* anglicky or se zapisuje || a vrací true tehdy a jen tehdy, pokud alespoň jeden jeho operand má hodnotu true.

```
if( x == 0 || y == 0 ) {
      System.out.println( ” At least one of the numbers x,
      y is equal 0” );
}
```

Oba tyto operátory používají tzv. **zkrácené vyhodnocování**, tj. druhý operand se vyhodnotí pouze v případě, že po vyhodnocení prvního operandu není známa hodnota celého výrazu. Např. má-li v logickém součinu první operand hodnotu false, druhý operand se nevyhodnocuje a hodnota celého výrazu je false. Protože se tyto operátory používají často v podmínkách, říká se jim podmínkové.

Kromě podmínkových operátorů má Java také operátory, které vyhodnocují vždy oba operandy. Zapisují se & (logický součin) a | (logický součet). Lze je použít na stejném místě jako podmínkové operátory.

```
boolean b1 = x > 0 & y == 1;
boolean b2 = x <= 0 | y <= 0;
```

Kombinujeme-li logický součin s logickým součtem, je potřeba mít na paměti, že logický součin má vyšší prioritu než logický součet:

```
if( x == 0 || y > 0 && z > 0 ) {
      System.out.println( ”x is zero or y and z are positive ”
);
}
```

3.3. Přiřazovací operátory

Jeden z přiřazovacích operátorů jsme již poznali, operátor =. Další přiřazovací operátory nám umožňují provést s proměnnou nějakou aritmetickou operaci. Např. operátor += přičte k proměnné hodnotu druhého operandu.

```
x += 5;
```

Dalšími přiřazovacími operátory jsou −=, *=, /=, %=. Každý z nich je zápisem příslušné operace s proměnnou na levé straně. Např. operátor %= provede operaci modulo:

```
x %= 6;  // stejné jako x = x % 6
```

3.4. Priorita provádění operací

Všechny operátory vrací hodnotu, která je výsledkem příslušné operace. Mají stejnou prioritu a asociují zprava doleva. V příkazu

```
int y = x + = 1;
```

se nejprve provede operátor += a teprve pak =. Operátor += vrátí hodnotu x po přičtení jedničky. Tato hodnota se použije jako hodnota druhého operandu operátoru =.

Probrané operátory si můžeme seřadit podle priority (od nejvyšší k nejnižší):

- inkrementace (++), dekrementace (−−)
- přetypování
- násobení (*), dělení (/), modulo (%)
- sčítání (+), odčítání (−)
- přiřazovací operátory (=, +=, −=, *=, /=, %=)

Priorita nám říká, jak silně se operátory váží na operandy. Např. přetypování má vyšší prioritu než násobení, proto se ve výrazu (int) d * 2 provede nejprve přetypování a až potom násobení.

```
double d = 5.8;
int i = (int) d * 2;  // same like ((int) d) * 2
System.out.println( i );
```

Tento kód tedy vytiskne hodnotu 10.

Přiřazovací operátory asociují zprava doleva a aritmetické operátory (sčítání, odčítání, násobení, dělení, modulo) zleva doprava.

Všechny binární operátory vyhodnocují operandy ve stejném pořadí: nejprve levý a pak pravý. Toto pořadí je významné, pokud má vyhodnocení operandů vedlejší efekt.

```
int x = 0;
int y = x + x++;
```

Na druhém řádku se nejprve vyhodnotí operátor +. Jeho levý operand má hodnotu 0, pravý operand má také hodnotu 0, operátor tedy vrátí 0 a ta se přiřadí do y. Při vyhodnocování pravého operandu dojde ke zvýšení proměnné x o 1 (vyhodnocení má vedlejší efekt). Prohodíme-li pořadí operandů, přiřadí se do y hodnota 1.

```
int x = 0;
int y = x++ + x;
```

3.5. Relační operátory a operátory rovnosti

Pro zápis podmínky používáme tzv. **relační operátory** a **operátory rovnosti**. Relační operátory jsou:

- menší než (<)
- větší než (>)
- menší nebo rovno (<=)
- větší nebo rovno (>=)

Operátory rovnosti jsou:

- rovná se (==)
- nerovná se (!=)

4. Základní programovací struktury

4.1. If

K větvení programu slouží příkazy if a switch. Příkaz if umožňuje větvit program na základě nějaké podmínky. Začíná klíčovým slovem if, za nímž je v závorkách výraz typu boolean. Výraz typu boolean je výraz, jehož hodnota je true nebo false. Dále následuje příkaz. Při provádění se vyhodnotí výraz v závorkách a má-li hodnotu true (podmínka je splněna), provede se příkaz. V příkladu níže se testuje zda-li je proměnná x rovna nule. V případě, že bude rovna nule, přiřadí se proměnné x hodnota 2.

```
if (x==0) x=2;
```

Pro zápis podmínky používáme relační operátory a operátory rovnosti.

Příkaz if může obsahovat větev else, která se provede, pokud podmínka za if není splněna. Pokud podmínka není splněna a větev else chybí, neprovede se nic (bude se pokračovat za příkazem if).

```
if( x == 0 )
      System.out.println( ” Can not be divided by zero!” );
else
      z=5/x;
```

Chceme-li zapsat za podmínku více příkazů, použijeme blok. Blok začíná otevírací složenou závorkou { a končí zavírací složenou závorkou }. Blok je v Javě příkaz, takže jej můžeme použít všude tam, kde se může vyskytovat příkaz.

```
if( x == y ) {
      x++;
      y--;
}
```

Blok omezuje platnost deklarace proměnné. Každá deklarace je platná jen do konce bloku, v němž je uvedena. Říkáme, že je v tomto bloku **lokální**.

```
if( x == y ) {
    int z = y;
 }  // tato závorka ukončuje platnost deklarace proměnné z
 // zde již z nelze použít
```

V závorkách za if můžeme použít proměnnou typu boolean.

```
// In the month variable we have the serial number of the month
      boolean isMay = (month == 5);
      if( isMay ) {
            System.out.println( ”time for love” );
}
```

Pro zápis opačné podmínky používáme operátor logické negace (!).

```
boolean isHere = true;
if( ! isHere ) {
      System.out.println( ”Is not here” );
}
```

Potřebujeme-li rozvětvit program např. podle hodnoty celočíselné proměnné, můžeme použít zřetězení příkazů if.

```
if( x == 1 ) {
      System.out.println( ”one” );
} else if( x == 2 ) {
      System.out.println( ”two” );
} else if( x == 3 ) {
      System.out.println( ”three” );
}
```

4.2. Switch

Takto zřetězené if (jako v minulém příkladu) můžeme někdy nahradit příkazem switch. Jeho zápis začíná klíčovým slovem switch. Za ním je v závorkách výraz typu int nebo String (nebo typu, který lze zkonvertovat na int) a dále blok, v němž je libovolný počet návěští case. Za každým case je uvedena konstanta (nebo konstantní výraz, což je výraz, jehož hodnota je známa při překladu), dvojtečka a posloupnost příkazů.

```
switch( x ) {
      case 1:
            System.out.println( ”one” );
            break;
      case 2:
            System.out.println( ”two” );
            break;
      case 3:
            System.out.println( ”three” );
}
```

Při provádění se vyhodnotí výraz za klíčovým slovem switch a jeho hodnota se začne srovnávat s hodnotami uvedenými za case (v pořadí, v němž jsou uvedeny). Jakmile dojde ke shodě, začnou se provádět příkazy. Provádění příkazů končí příkazem break. Pokud příkaz break chybí, provedou se všechny příkazy až do konce příkazu switch. Příkaz switch může obsahovat větev default, která se provede, pokud nedošlo ke shodě u žádného návěští case.

5. Cykly

Cykly slouží k opakovanému provádění příkazů.

5.1. While

Cyklus while začíná klíčovým slovem while, za kterým následuje v závorkách podmínka a dále tělo cyklu. Tělem cyklu je buď příkaz, nebo blok. V rámci cyklu while se vyhodnotí se podmínka a je-li splněna, provede se tělo cyklu. Poté se znovu vyhodnotí podmínka a je-li splněna, opět se provede tělo cyklu, atd. Pokud podmínka není splněna, pokračuje se příkazy za cyklem. Jestliže na začátku není podmínka splněna, tělo cyklu se neprovede ani jednou. Cyklus while je tedy cyklus s počtem opakování 0 nebo více.

Příklad syntaxe:

```
While(condition) {
      // body
}
```

Konkrétní příklad

```
int x = 5;
while( x > 0 ) {  // Do until x is greater than zero
      System.out.println( x );
      x --;
}
```

Tento příklad vypíše čísla od 5 do 1. Proběhne tedy 5x.

5.2. Do while

Cyklus do while začíná klíčovým slovem do, za kterým je tělo cyklu, následuje klíčové slovo while a podmínka viz příklad syntaxe. Provádí se takto: nejprve se provede tělo cyklu, pak se vyhodnotí podmínka a je-li splněna, znovu se provede tělo cyklu, vyhodnotí se podmínka, atd. Není-li podmínka splněna, pokračuje se za cyklem. Tělo cyklu se tedy provede vždy alespoň jednou. Počet opakování je tedy 1 nebo více.

Příklad syntaxe:

```
do {
      // body
} while (condition);
```

Konkrétní příklad

```
do {
      System.out.println( x );
      x --;
} while (x > 0);
```

5.3. For

Cyklus for má tvar: for *(řídící proměnná cyklu; podminka; prikaz)* Provádí se takto: nejprve se provede inicializace řídící proměnné cyklu, pak se vyhodnotí podmínka a je-li splněna, provede se tělo cyklu. Poté se provede příkaz, znovu se vyhodnotí podmínka a je-li splněna, opět se provede tělo cyklu, atd. Inicializace řídící proměnné cyklu se tedy provede pouze jednou na začátku. Pokud na začátku není podmínka splněna, tělo cyklu se neprovede ani jednou.

Příklad syntaxe:

```
for (cycle variable; condition; command){
      // body
}
```

Konkrétní příklad

```
int a;
for( a = 1; a < 10; a++ ) {
      System.out.println( a );
}
```

Řídící proměnná cyklu může být v rámci cyklu nově deklarována. Tato deklarace je platná pouze v daném cyklu (tj. v hlavičce a těle cyklu). Říkáme, že proměnná je v tomto cyklu lokální.

```
for( int a = 1; a <= 10; a++ ) {
      System.out.println( a * a );
}
```

Libovolná z částí řídící proměnná cyklu, podmínka, příkaz může chybět. Chybí-li podmínka, jde o nekonečný cyklus (podmínka je stále splněna). Pokud není uvedena řídící proměnná cyklu nebo příkaz, neprovede se v daný okamžik žádný příkaz.

5.4. Break a continue

V těle cyklu můžeme používat příkazy break a continue. Příkaz break okamžitě ukončí provádění cyklu. Pokračovat se bude příkazy za cyklem.

```
int s = 100;
while( s > 0 ) {
      int n = sc.nextInt();
      if( n == 0 ) {
            break;
      }
      s -= n;
      System.out.println( s );
}
// Here will continue after the break executed
```

Příkaz continue ukončí provádění těla cyklu a přejde na vyhodnocení podmínky cyklu.

```
int s = 0;
do {
      int n = sc.nextInt();
      if( n == 0 ) {
            continue;
      }
      s += n;
      System.out.println( s );
      // here goes continue
} while( s < 100 );
```

6. Statické metody

Doposud jsme zapisovali celý program do metody main. Pokud bylo třeba provést stejnou posloupnost příkazů na více místech, museli jsme tyto příkazy zopakovat. Tomu se lze vyhnout. Metody nám umožňují členit kód do logických celků a tyto celky opakovaně využívat. V této kapitole budeme pod pojmem metoda rozumět statickou metodu. Jednu statickou metodu již známe. Je to metoda main. Kromě metody main můžeme ve třídě deklarovat i další metody, např. metodu pro tisk informací o programu.

```
class MainClass {
      static void printInfo() {
            System.out.println( "Version: 1.0" );
            System.out.println( " Autor: 007" );
      }
      public static void main( String[] args ) {
            printInfo ();
      }
}
```

Deklarace statické metody začíná klíčovým slovem static. (viz příklad výše) Za ním následuje návratový typ a jméno metody. Návratovým typem může být libovolný javovský typ. Pokud metoda nevrací žádnou hodnotu, uvedeme jako návratový typ void. Jméno metody obvykle začíná malým písmenem. Pokud se jméno skládá z více slov, oddělujeme slova tak, že první písmena dalších slov píšeme velká. Např. spoctiPolomerKruzniceOpsane. Není zvykem používat ve jméně metody podtržítka. Za jménem metody je v závorkách seznam parametrů. Seznam parametrů je tvořen deklaracemi proměnných. Oddělovačem deklarací v seznamu parametrů je čárka.

6.1. Volání metody

V místě, kde chceme metodu provést, zapíšeme volání metody. Volání metody se skládá ze jména metody a seznamu parametrů. Návratovou hodnotou metody bude hodnota výrazu, který je uveden za příkazem return.

Na pořadí, v němž metody deklarujeme, nezáleží. Lze volat i metodu, která je deklarována později.

```
class MainClass {
      public static void main( String[] args ) {
            Scanner sc = new Scanner( System.in );
            int x = sc.nextInt();
            long factorial = countFactorial( x );
            System.out.printf( "%d! = %d%n", x, faktorial );
      }
      static long countFactorial ( int n ) {
            long fact = 1;
            for( ; n > 1; n-- ) {
            fact *= n;
      }
      return fact;
      }
}
```

V metodě typu void lze použít příkaz return bez parametrů. Toho se využívá pro předčasné ukončení metody.

```
// print rectangle a x b from @
static void printRectangle ( int a, int b ) {
      // The minimum rectangle side size is 2
      if( a < 2 || b < 2 ) {
            return;
      }
      for( ; a > 0; a-- ) {
            for( int i = 0; i < b; i++ ) {
                  System.out.print( '@' );
            }
            System.out.pritln();
      }
}
```

Příkaz return se může v metodě vyskytovat vícekrát. Provede se však vždy jen jednou jako poslední příkaz metody. Jeho provedení způsobí okamžité ukončení metody.

```
static boolean isPrimeNumber ( int n ) {
      if( n == 2 ) {  // 2 prime number
            return true;
      }
      if( n % 2 == 0 ) {
            // Even number is not a prime number (except 2)
            return false;
      }
      int sqrt = (int) Math.sqrt( n );
      for( int i = 3; i <= sqrt; i += 2 ) {
            if( n % i == 0 ) {
                  // We found a divisor, so n is not a
                  prime number
                  return false;
            }
      }
      return true;
}
```

7. Instanční proměnné

Instančním proměnným říkáme instanční atributy nebo zkráceně atributy (angl. instance attributes nebo fields). Statické metody již známe. Deklarují se pomocí klíčového slova static.

Instanční metody se deklarují podobně jako statické. Na rozdíl od statických metod však jejich hlavičky neobsahují klíčové slovo static. Instanční metody často pracují s instančními atributy.

7.1. Pole

Pole umožňuje pracovat s více hodnotami stejného typu. Pro uložení např. dvaceti hodnot typu int můžeme buď zavést dvacet proměnných, nebo vytvořit pole o dvaceti prvcích. V mnoha případech se s polem pracuje snadněji. Typ pole zapisujeme v Javě pomocí hranatých závorek. Deklarace proměnné typu pole položek typu int vypadá takto:

```
int[] p;
```

Proměnná typu pole je tzv. reference. Bude obsahovat odkaz (referenci) na pole. Samotná deklarace pole nevytváří. Pole můžeme vytvořit pomocí klíčového slova new:

```
p = new int[6];
```

V tomto případě jsme vytvořili pole o šesti prvcích typu int. Potřebujeme-li počet prvků pole, použijeme názevPole.length. V našem případě tedy

```
p.length
```

Hodnota této proměnné je nastavena při vytvoření pole a nelze ji změnit (je pouze pro čtení).

```
System.out.printf( ”array p has %d elemets%n”, p.length );
```

K jednotlivým prvkům pole přistupujeme pomocí indexů, které zapisujeme do hranatých závorek:

```
p[1] = 5;
```

Indexy začínají vždy od nuly, První číslo bude mít index 0, druhé číslo bude mít index 1, třetí číslo bude mít index 2 atd Platnost indexu se kontroluje za běhu programu. Použití neplatného indexu způsobí běhovou chybu. Po vytvoření jsou prvky pole inicializovány na hodnoty, jejichž vnitřní reprezentace je 0. U číselných typů je to 0, u typu boolean je to false a u typu char je to znak na pozici 0 v tabulce Unicode. Pro načtení hodnot do pole používáme obvykle cyklus for:

```
int[] p = new int[10];
      for( int i = 0; i < a.length; i++ ) {
            p[i] = i;
}
```

Výpis prvků pole provedeme nejčastěji opět cyklem for:

```
for( int i = 0; i < a.length; i++ ) {
      System.out.println( a[i] );
}
```

Délka pole musí být nezáporná. Pokus vytvořit pole záporné délky způsobí chybu. Vytvoření pole lze spojit s inicializací. V takovém případě se nepoužívá new:

```
int[]numbers = { 3, 5, 6, 7};
```

Velikost pole je dána počtem hodnot ve složených závorkách. Pole může být parametrem metody a může být i návratovým typem.

7.2. Vícerozměrná pole

Doposud jsme pro určení prvku v poli používali jeden index. Takovému poli říkáme jednorozměrné. Java umožňuje deklarovat a vytvářet i vícerozměrná pole. Např. deklarace dvojrozměrného pole položek typu int vypadá takto:

```
int[][] p;
```

Vícerozměrné pole v Javě je pole polí. Proměnná p je reference na pole, jehož prvky jsou jednorozměrná pole celých čísel. Pole vytvoříme pomocí klíčového slova new:

```
p = new int[2][3];
```

K prvkům pole přistupujeme pomocí indexů:

```
p[0][1] = 1;
```

Počet prvků pole je v proměnné length daného pole.

```
System.out.printf( "pole p má d řádkůn", p.length );
System.out.printf( "první řádek má d sloupcůn", p[0].length );
```

Při práci s vícerozměrnými poli používáme obvykle vnořené cykly for. Např. inicializaci prvků pole p na hodnotu 1 můžeme provést takto:

```
for( int i = 0; i < p.length; i++ ) {
  for( int j = 0; j < p[i].length; j++ ) {
       p[i][j] = 1;
  }
}
```

Vícerozměrná pole je možné vytvářet postupně. Dílčí pole pak mohou mít různý počet prvků. Dvojrozměrné pole tedy nemusí být nutně obdélníkové.

8. Třídy

Třída je forma, která popisuje jednoznačně ohraničený soubor (typů) dat a operací nad nimi. Pomocí třídy pak vytváříme instance, jednotlivé objekty, které obsahují samotná data (nad kterými jsou volány příslušné operace).

Příklad: Mějme třídu Auto, tato třída popisuje, že auto má značku, typ, stáří a najeto a obsahuje operaci informaceOVozidle(), která vypíše zadané značku typ, stáří a nejeto . Pomocí této šablony – třídy – pak vytváříme jednotlivé instance, v reálném životě bychom je popsali jako konkrétní vozidla.

Každé nově vytvořené instanci (vozidlu) v programu přiřadíme data (značka, typ, stáří a najeto). Když později zavoláme operaci informaceOVozidle () nad tímto objektem (instancí), tak nám vypíše hlášku specifickou pro dané vozidlo.

8.1. Deklarace třídy

Pro deklaraci třídy používáme klíčové slovo class, před kterým je specifikátor přístupu, a za kterým následuje název třídy. Samotné tělo třídy je uzavřeno v bloku (složené závorky). Pokud se název skládá z více slov, zpravidla první písmeno každého slova píšeme velké (např. VypisInfo). Nepoužíváme podtržítka. Ve třídě můžeme deklarovat proměnné a metody. Proměnné i metody mohou být buď statické, nebo instanční.

```
class Cat {
  int weight;  // instance atribut
  int age;
  void showInfo() {  // Instance method
        System.out.println( info );
  }
}
```

Od dané třídy můžeme vytvářet instance (říkáme jim též objekty, angl. instances či objects). Třída je šablona, která říká, jak budou objekty vypadat, tj. jaké budou mít atributy a metody. V našem případě bude mít každá instance třídy Kočka dvě proměnné typu int. Deklarací proměnné typu Kočka zavedeme proměnnou, do níž můžeme uložit referenci (odkaz) na instanci třídy MojePrvni.

```
Cat  v;
```

Protože proměnné typu třída odkazují na objekty, nazývají se referenční proměnné. Každá referenční proměnná zabírá v paměti stejný prostor: 32 bitů v 32-bitové JVM a

obvykle 64 bitů v 64-bitové JVM. Naproti tomu objekty různých typů mají zpravidla různou velikost. Velikost objektu je dána jeho atributy. Objekty vytváříme pomocí klíčového slova new:

```
V = new Cat();
```

Po provedení tohoto příkazu bude proměnná v obsahovat odkaz na instanci třídy Kocka. K atributům a metodám přistupujeme pomocí tečky. Volání metody zapisujeme pomocí jména metody a kulatých závorek:

```
v.info = 1;
v. showInfo ();
```

Od jedné třídy můžeme vytvořit libovolné množství instancí. Tyto instance jsou na sobě nezávislé.

```
Cat v2 = new Cat();
v2. showInfo = 2;
v2. showInfo ();
```

Instanční metody slouží k provádění operací nad objekty daného typu. Např. ve třídě MojePrvni můžeme deklarovat metodu, která zvýší hodnotu atributu x o 1:

```
class My {
  int x;
  void IncreaseA () {
        a++;
  }
}
```

Instanční metody, stejně jako metody třídní, mohou mít parametry a vracet hodnotu. Parametry a návratovou hodnotu stanovíme v deklaraci metody.

```
class My {
  int x;
  // Adds dx to x and returns a new x value
  int moveX( int dx ) {
        x += dx;
     return x;
  }
}
```

9. Specifikátory přístupu

Pro specifikace práv přístupu k jednotlivým třídám, jejich operacím a proměnným používáme specifikátory přístupu. Jejich význam je především v zakrývání implementačních detailů, které nemá (nesmí) uživatel vidět a případně je používat.

Public	Z jakékoliv třídy.
Private	Pouze uvnitř dané třídy, žádný přístup z vnějšku.
Protected	Z jakékoliv třídy stejného balíku, případně z potomka třídy kdekoliv.
žádný (package friendly)	Z kterékoliv třídy stejného balíku.

9.1. Konstruktory

Vytvoříme-li novou instanci třídy Kočka, budou všechny její instanční proměnné inicializovány na nulové hodnoty. Pokud budeme chtít u nově vytvořené kočky zadat její stáří, musíme použít speciální metodu: konstruktor, která se jmenuje stejně jako třída a u níž se neuvádí typ návratové hodnoty. V konstruktoru můžeme nastavit počáteční hodnoty instančních proměnných.

Jako příklad máme zaměstnance, který má svůj věk a plat. Obsahuje dále metodu IntroduceYourself.

```
class Employee {
  public Employee (int age, int wage) {
     this.age = age;
     this.wage = wage;
  }
  private int age = 1;
  public int getAge () { return age; }
  public void setAge(int age) { this. age = age; }
  private int wage = 1;

  public int getWage() { return wage; }
  public void setWage(int wage) { this.wage = wage; }
  public void introduceYourself(){
       System.out.println("My age a wage are "
       + age + "years"+ wage + "Euros");
  }
  public static void main(String[] args) {
    Employee employee = new employee (30,100);
  }
}
```

9.2. Dědičnost

Pomocí dědičnosti můžeme vytvářet hierarchie tříd, ve kterých můžeme o libovolném uzlu říct, že je speciálním případem libovolného ze svých předků. V normálním světě bychom řekli, že židle je typem nábytku, letadlo je typem dopravního prostředku a dopravní prostředek je typem stroje. Pozor ale nemůžeme dědit židli od zvířat, protože mají také 4 nohy!

V Javě pro vytvoření podtypu v hlavičce třídy ihned po jejím názvu uvádíme klíčové slovo extends a název rozšiřované třídy. Takto vytvořená třída zdědí všechny nesoukromé (včetně package friendly, je-li rozšiřující třída ve stejném balíčku) metody a třídní proměnné předka, které mohou být znova deklarovány a překryty.

Naopak třída nezdědí soukromé a statické metody svého předchůdce, protože ty se vztahují pouze ke konkrétní třídě předka. Koncové (final) metody označené jako potomek, třída zdědí, ale nemůže je překrýt. Každý objekt potomka můžeme přetypovat na předka.

9.3. Super

Při voláních metod podtypů často narazíme na to, že nechceme celou metodu překrýt, pouze k ní chceme přidat další funkcionalitu. V tento okamžik můžeme zavolat super.jmenoMetody(), čímž zavoláme funkcionalitu předka. Také můžeme volat konstruktor předka voláním super() – toto volání musí být v rámci konstruktoru potomka uvedeno nejdříve.

Jako příklad jsme vytvořili třídu ředitel, která je odvozena od třídy zaměstananec. Ovšem to neznamená, že by tato třída měla automaticky přístup ke všem soukromým proměnným a metodám původní třídy. Viz. Specifikátory přístupu.

Volání konstruktoru nadřazené třídy se provádí pomocí klíčového slova super a musí být prvním příkazem v těle konstruktoru. Pokud, že konstruktor nadřazené třídy nezavoláme, zařadí se do programu jeho volání automaticky - v takovém případě se volá tzv. implicitní konstruktor, tedy konstruktor bez parametrů.

```
class Director extends Employee {
  public Director(int age, int wage)
  {
        super(age,wage);
  }
  public static void main(String[] args) {
        Employee k = new Director(30, 50);
  }
}
```

10. Polymorfismus

Můžeme jakoukoliv metodu překrýt v potomkovi vlastní implementací. Když nad daným objektem tuto metodu zavoláme, dojde vždy k vykonání překrývajícího kódu. A to bez ohledu na to, jestli k metodě přistupujeme pomocí reference na objekt předka nebo na objekt potomka (jehož třída obsahuje ono překrytí).

Této vlastnosti je dosaženo pomocí pozdní vazby (late binding), kdy je typ objektu, na němž bude metoda volána, rozhodnut až za běhu programu, nikoliv během jeho kompilace.

Toto ovšem platí pouze pro překrývání metod, nikoliv pro jejich přetěžování. Pokud budeme mít na daném objektu dvě metody, které se budou lišit pouze parametrem (jedna pro předka, druhá pro potomka), tak bude volána vždy ta metoda, která má v parametru stejný typ, jako je aktuální reference na objekt. Volání přetížených metod je totiž rozhodováno v době překladu, kdy ještě nemusí být jasné, jestli bude reference ukazovat na objekt předka nebo potomka.

OBJEKTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ JAVA

1. Třídy

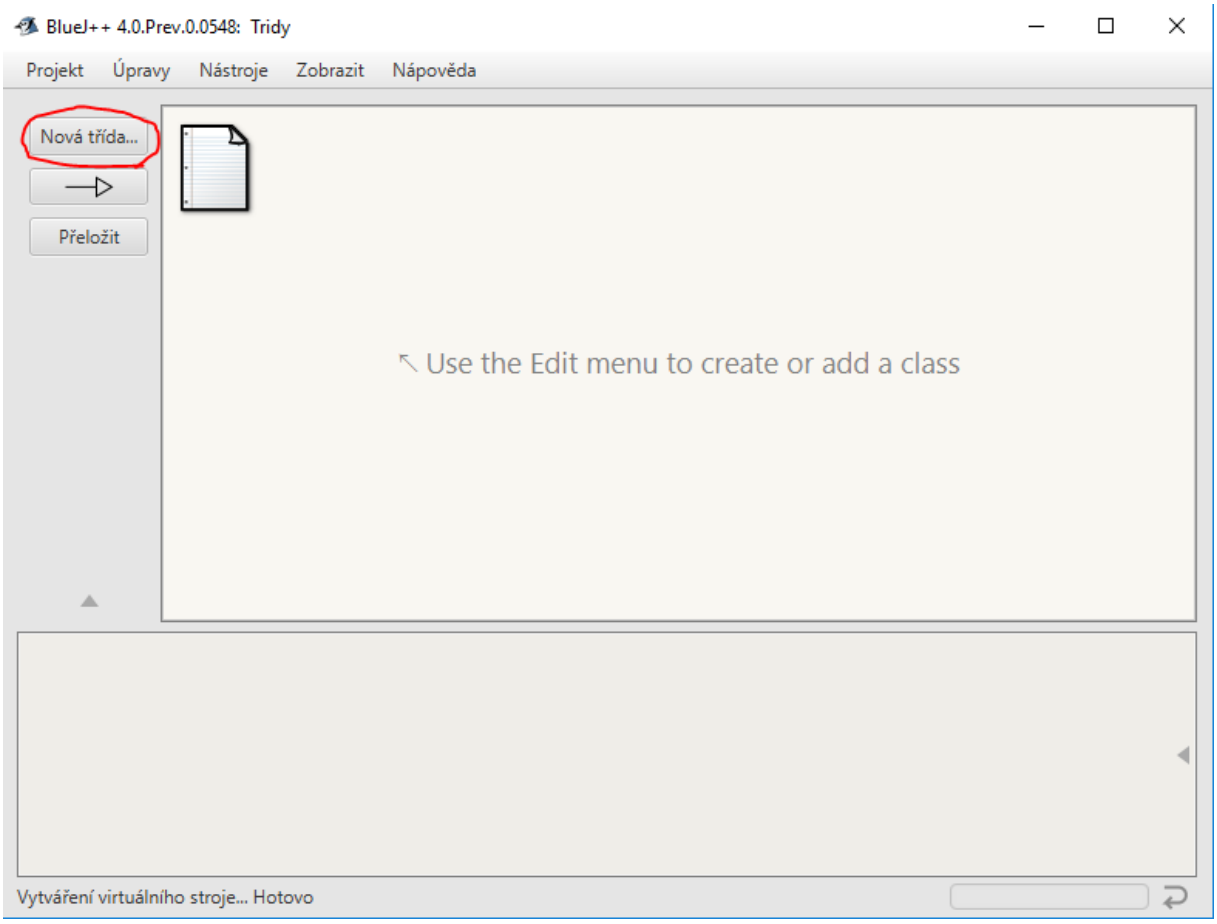
V reálném životě je například slovo židle názvem kusu nábytku, jehož funkcí je, že se dá na ní sedět. Je to tedy popis objektu, který je charakterizován pomocí různých vlastností, či funkcí. Analogií v programování je třída. Třída seskupuje objekty se nějakými společnými vlastnostmi. V reálném životě existuje velké množství rozdílných židlí, které se mohou lišit třeba v materiálu či barvě. V programování konkrétní židli odpovídá instance příslušné třídy, někdy se též nazývá objekt. Třídu tak můžeme přirovnat k formě, do které se odlévá konkrétní věc – instance. Běžně mohou mít třídy libovolný počet instancí.

Jednotlivé objekty mezi sebou mohou komunikovat, navzájem si posílat různé zprávy, ve kterých se mohou žádat o různé informace, nebo služby. Příkladem může být kalkulačka, kterou můžeme požádat o spoustu úkonů, například sečtení dvou čísel. Každá z funkcí této kalkulačky je odborně nazývána metoda. Žádost o využití konkrétní metody se nazývá volání metody. Metoda je tedy část programu, kterou instance pouští jako reakce na volání metody (obdržení zprávy). Tvůrce metody v metodě definuje, jak má objekt na příslušnou zprávu zareagovat.

Celý objektově orientovaný program Pecinovský (2004) popisuje jako v nějakém programovacím jazyce zapsaný popis použitých tříd, objektů a zpráv, které si tyto objekty posílají, doplněný u složitějších programů ještě o popis umístění programů na jednotlivých počítačích a jejich svěření do správy příslušných služebních programů. (Např. OS, nebo aplikačních serverů).

1.1. Tvorba tříd

Nyní již přikročíme k tvorbě samotných tříd. Tedy jakýchsi vzorů či forem, podle kterých vznikají konkrétní instance (objekty). V programu BlueJ, který budeme používat po celou dobu tohoto kurzu, zvolíme nový projekt. Následně zvolíme tlačítko na levé straně. Tak jak to ukazuje obrázek níže.



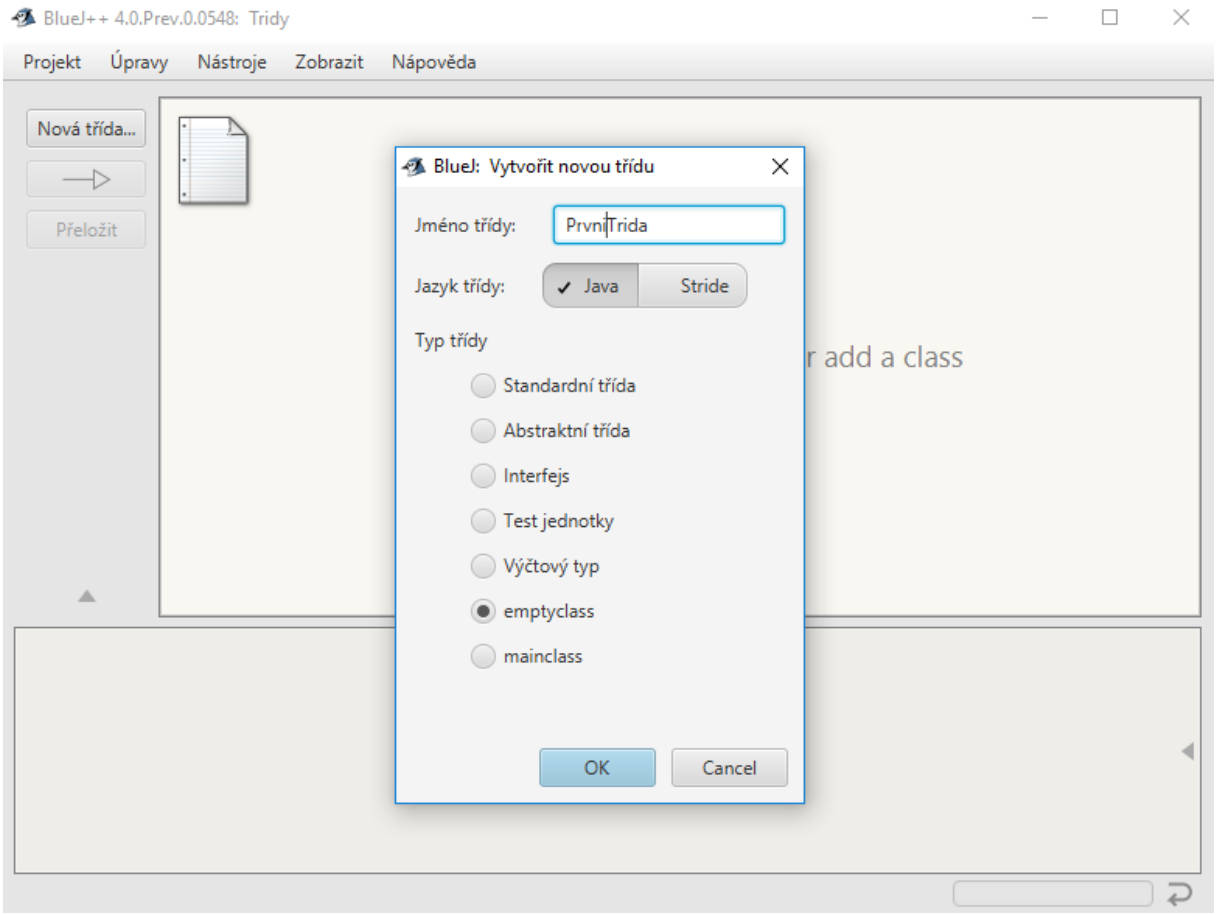
V následujícím okně zvolíme nejjednodušší možnou definici třídy emptyclass – prázdnou třídu. Dále zvolíme název třídy. V našem projektu jsme zvolili název PrvniTrida.

Název třídy se řídí několika pravidly:

Pravidla pro tvorbu názvu identifikátorů

Odkazy, třídy a další, musí být pro snadnou identifikaci pojmenovány. Těmto názvům říkáme identifikátory. Tyto identifikátory musí splňovat několik podmínek:

- Smí obsahovat libovolné znaky, které jsou obsaženy v sadě UNICODE
- Nesmí se používat mezery
- Je zvykem psát víceslovní názvy tříd bez mezer. Jednotlivá slova pak začínat velkými písmeny třeba: MojePrvniTrida
- Velká a malá písmena se považují za rozdílná
- Délka je neomezená
- Nesmí začínat číslicí
- Nesmějí být shodné s klíčovými slovy



Po vytvoření a přeložení máme první třídu. Pokud se podíváme do složky, ve které máme uložený celý projekt. Zjistíme, že ve složce přibilo několik souborů

PrvniTrida.java Tento soubor označujeme, jako zdrojový soubor. Do něj budeme zapisovat program, popisující chování třídy a tedy i jejích instancí. Tento soubor je pouze textovým souborem. Můžete jej tak editovat v libovolném textovém editoru.

PrvniTrida.class V tomto souboru je uložen přeložený bajtkód

PrvniTrida.ctxt Pomocný soubor prostředí BlueJ. Pokud tento soubor smažeme BlueJ si jej při příštím překladu vytvoří znovu

1.2. Práce s třídou

Po dvojkliku na třídu se nám otevře okno, ve kterém může psát, či editovat zdrojový kód třídy. Viz obrázek níže:

V textovém okně uvidíme text – který můžeme nazvat prázdná definice třídy. Tento text za nás automatiky vygeneroval BlueJ v momentě vzniku třídy.

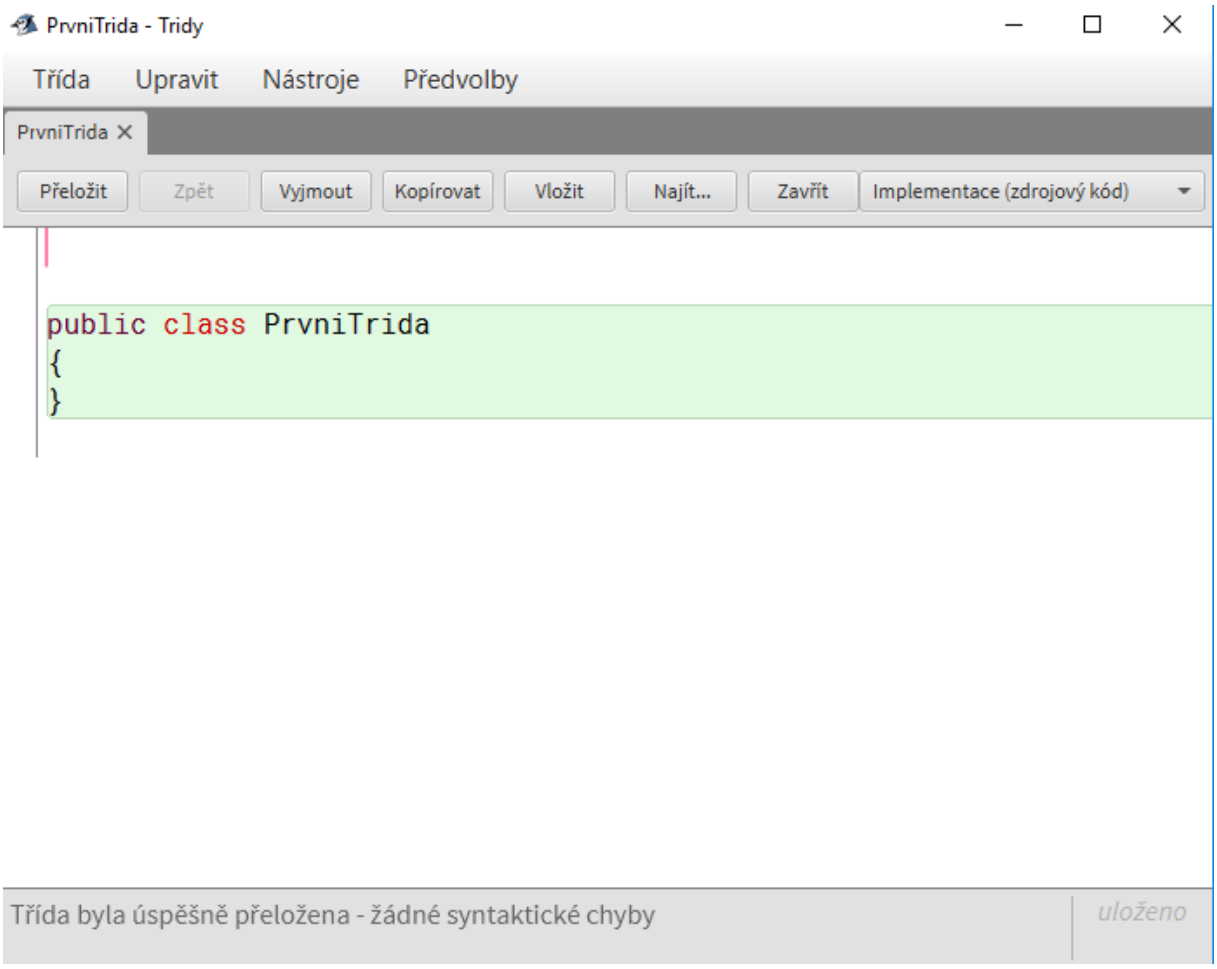
Tento text obsahuje slova

Public Klíčové slovo, které identifikuje, že s třídou může pracovat kdokoliv

Class Klíčové slovo, které oznamuje, že následně bude definice třídy

PrvniTrida Název třídy (její identifikátor)

Následuje samotné tělo třídy, které je uzavřeno v složených závorkách. V našem případě tělo třídy zatím nic neobsahuje.



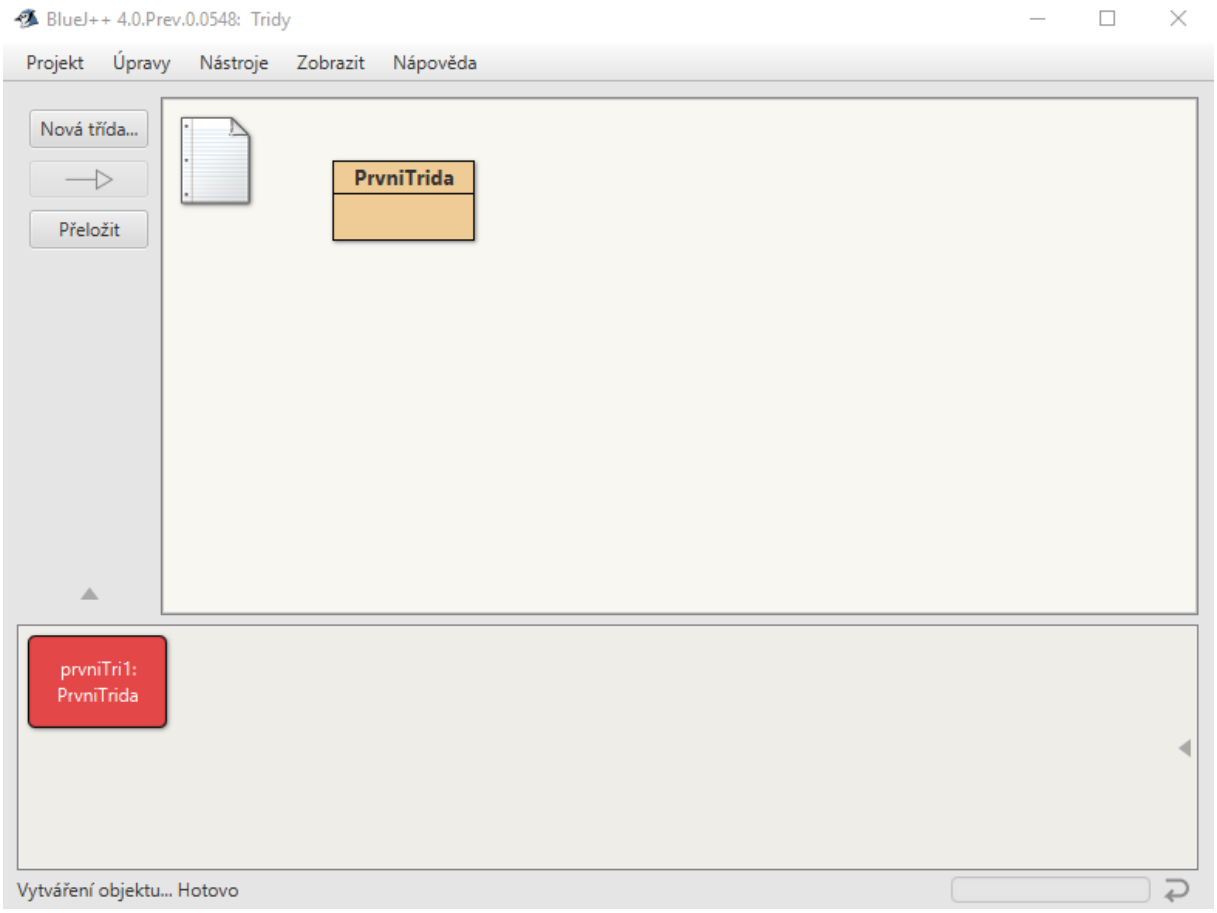
1.3. Vytvoření první instance

Zavřeme textové okno. Pokud klikneme pravým tlačítkem na naši první třídu. Pozor! Dotyčná třída musí být zkompilovaná! V rozbalovací nabídce vybereme příkaz new PrvniTrida(). BlueJ se nás následně zeptá na název právě vznikající instance. Zároveň nám nabídne jím před vybraný název. Který stačí pouze schválit. Čímž jsme vytvořili první třídu. Jak je vidět na obrázku níže.

Vybráním příkaz new PrvniTrida() vytváříme instanci třídy tak, že zasíláme zprávu sestavenou z klíčového slova new, následně uvedeme název třídy, ze které chceme instanci vytvořit dvojici kulatých zvorek. Tím voláme speciální metodu, která se jmenuje konstruktor. Konstruktor požadovanou instanci vytvoří a vrátí nám odkaz, přes který se budeme na vytvořenou instanci obracet.

V našem případě jsme žádnou metodu konstruktoru nevytvořili. Konstruktor musí mít ale každá třída! V případě, kdy nevytvoříme žádný konstruktor, překladač automaticky vytvoří nejjednodušší konstruktor, který označujeme – implicitní konstruktor. V momentě, kdy vytvoříme ve třídě první konstruktor, překladač přestává s tvorbou implicitního konstruktoru.

Vytvořená instance třídy je vidět na obrázku níže. Je v dolní části, v oblasti, kterou nazýváme zásobník odkazů. Instance třídy má červenou barvu.



1.4. Odstranění třídy

Velmi jednoduše. Klikneme pravým tlačítkem na danou třídu a zvolíme možnost odstranit. Po potvrzení je třída opravdu smazána. Samotná instance třídy ale zůstala. Žádost o smazání třídy je vlastně žádostí o smazání příslušných souborů z disku.

Samotná instance třídy je uložená v operační paměti. Pokud chceme odstranit i instanci třídy, musíme restartovat virtuální stroj.

1.5. Restartování virtuálního stroje

Restartováním virtuálního stroje smažeme všechny odkazy v zásobníku odkazů. Smažeme tedy všechny instance. Dosáhneme toho buď klávesovou zkratkou Ctrl + shift + r anebo kliknutím pravého tlačítka myši na obdélníček vpravo dole a vybráním příkazu: Restartovat virtuální stroj.

2. Konstruktory

V předcházející kapitole za nás překladač vytvořil implicitní konstruktor. Nyní si ukážeme, jak tvořit vlastní konstruktory.

2.1. Bezparametrický konstruktor

Vytvořte si třídu student, podobné té, kterou jsme smazali. Nyní si vytvoříme bezparametrický konstruktor. Bezparametrický znamená, že pro svůj běh nevyžaduje žádné informace – parametry. Samotná deklarace třídy a konstruktoru by mohly vypadat na nějak takto. Samotný konstruktor je žlutý:

```
{
        int math =3;
              int english =3;
              int ICT =3;
        public Student ()
        {
              .........
        }
}
```

Nyní si projdeme význam jednotlivých částí textu.

Nejdříve jsme deklarovali třídu Student a proměnné matematiky, cestina a informatika.

Samotná hlavička konstruktoru vypadá dosti podobně, jako hlavička třídy. Jen je vynecháno klíčové slovo class. Název konstruktoru musí být stejný, jako název třídy, ve které je konstruktor uveden. Následují závorky, do kterých buď píší parametry, pak jde o parametrický konstruktor, pokud se do závorek nic nenapíše, půjde o bezparametrický konstruktor. Samotné tělo konstruktoru je v složených závorkách.

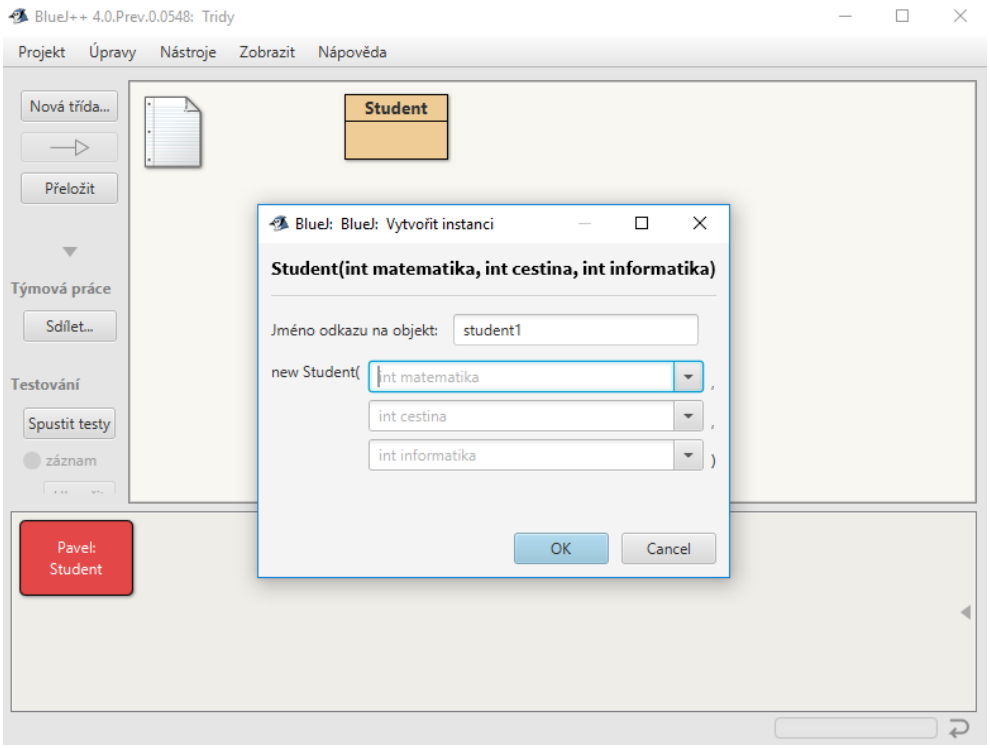
2.2. Konstruktor s parametry

Pokud bychom tvořili instance typu Student, každý ze studentů by měl ty samé známky. Vytvoříme proto tentokrát další konstruktor. Tentokrát již s parametry.

```
public class Student
{
      int math =3;
      int english =3;
      int ICT =3;
      public Student ()
      {
            .........
      }
      public Student (int math,int english,int ICT)
      {
            .........
      }
}
```

Nový parametrický konstruktor jsme žlutě zvýraznili. Všimněte si, že oba konstruktory mají stejný název. Překladač mezi nimi rozlišuje podle počtu a typu parametrů. Do parametrického konstruktoru, uvádíme nejdříve typ a následně identifikátor, jehož prostřednictvím se program na parametr v těle konstruktoru odvolává. Nemůžeme tedy vytvořit konstruktory se shodnými typy parametrů. Překladač dokonce nerozlišuje ani návratové hodnoty u jednotlivých konstruktorů. Využití více konstruktorů nazýváme přetěžování.

Nyní když budeme chtít vytvořit novou instanci – žáka pomocí parametrického konstruktoru, budeme dotázáni, na zadání parametrů typu integer tak, jak je tomu na obrázku níže.



2.3. This

Mnohé konstruktory jsou si navzájem podobné. Mnohdy se stane, že chceme využít v jednom konstruktoru jiný konstruktor. Tomuto opakovanému psaní těla konstruktoru se může vyhnout pomocí klíčového slova this, následovaného seznamem parametrů. Pozor volání jiného konstruktoru pomocí this musí být úplně prvním příkazem konstruktoru! Využití konstruktoru this je na dalším příkladu.

```
public class Student
{
      int math =3;
      int english =3;
      int ICT =3;
      public Student ()
      {
            this (0, 0, 0);
      }
      public Student (int math,int english,int ICT)
      {
            .....
      }
```

3. Metody

Metoda je specifický podprogram, který vykonává, nějakou specifickou funkci. A patří mezi nejčastěji používané nástroje (téměř) každého programovacího jazyka, tedy i jazyka Java. Metody bychom mohli přirovnat k nástrojům, kterými následně je vybavena každá instance třídy. Samotná metoda se skládá z několika částí:

- **Specifikátor přístupu** – který určuje, kdo všechno smí metodu volat. Nejčastěji se používá public a private. Specifikátor je nepovinný
- **Typ návratové hodnoty** – Povinný. Pokud metoda nic nevrací, uvádíme void
- **Název metody** – platí stejná pravidla, jaká byla uvedena u konstruktorů
- **Seznam parametrů metody** – také stejný princip jako u konstruktorů

Samotné tělo metody je uzavřeno do složených závorek. Kam můžeme psát jednotlivé příkazy. Pokud chceme, nemusíme do těla metody napsat nic.

```
public void Hello()
{
      System.out.println(”Hello”);
}
```

3.1. Metody vracející nějakou hodnotu

Pokud má metoda vracet nějakou hodnotu, musíme v hlavičce specifikovat její typ a v těle metody musíme uvést příkaz return, za kterým následují proměnná, kterou chceme vrátit. Po příkazu return bude metoda okamžitě ukončena. Nebude tedy vykonán kód, který v těle metody následuje po příkazu return. Příklad je uveden níže.

```
public double averageGrade ()
{
      double average= (math + english + ICT)/3;
      return average;
}
```

3.2. Volání metody

Samotným napsáním kódy se žádný kód neprovede. Provede se, až když jej zavoláme. Volání metody lze provést pouze z místa, ze kterého je metoda přístupná. Volání metody provádíme napsáním názvu metody a do závorek uvedeme případné parametry metody. Pokud je metoda uvedena v jiné třídě. Musíme nejdříve uvést jméno třídy, samotný název metody oddělujeme tečkou. Posíláme-li zprávu instanci, musíme nejdříve napsat odkaz na tuto instanci. U atributů je situace podobná.

3.3. Předávání parametrů metodám

Hodnoty primitivních typů jako jsou například znaky, logické hodnoty, nebo čísla se předávají tak, že hodnota se překopíruje do lokální proměnné metody

Hodnoty objektových typů se předávají odkazem. Tedy do lokální proměnné metody se pouze nakopíruje odkaz na objekt, ve kterém je skutečný parametr.

4. Statické atributy

Statické prvky patří třídě, ne instanci. Statické atributy označujeme slovem **static**. Díky tomu že patří třídě, k němu mají přístup všechny metody dané třídy. Statické atributy můžeme číst i v případě, že neexistuje instance třídy. Deklarace statického atributu je uvedena níže a je zažlucena.

```
class Group {
      private static int number = 15;
      public void New(int count) {
            number = number + count;
      }
}
```

4.1. Statické třídy

Metody třídy se volají na třídě. Velmi často jde o pomocné metody, které často používáme, ale nechceme speciálně kvůli tomu tvořit instanci. Jako příklad může sloužit statická metoda, která testuje, zda je zadané číslo kladné.

```
public static boolean isPlus(int number) {
      if (number >= 0) {
            return true;
      }
      return false;
}
```

Pozor! Díky tomu, že statická metoda náleží třídě, nemůžeme v ní přistupovat k žádným instančním atributům. Tyto atributy totiž neexistují v rámci třídy, ale instance.

4.2. Lokální proměnné

Občas potřebujeme v rámci metody něco zapamatovat. K tomuto účelu slouží lokální proměnné. Deklarujeme je uvnitř metody. Mimo metodu se k nim nedá přistupovat. Díky čemuž můžeme v jiné metodě definovat jinou lokální proměnnou se stejným názvem. V jejich deklaraci nepoužíváme modifikátory přístupu (např. public a private) ani static. Při jejich deklaraci jim musíme přiřadit nějako konkrétní hodnotu. V momentě opuštění metody se lokální proměnná zruší. Nelze v nich tedy uchovávat nic, co bychom potřebovali mezi různými metodami. Častým důvodem využívání lokálních proměnných bývá zpřehlednění programu a snížení počtu chyb zavlečených v důsledku opakovaného opisování složitých výrazů. Deklarace lokální proměnné je na příkladu níže.

```
public void totalPrice (int pieces)
{
      int totalPrice = pieces *15;
}
```

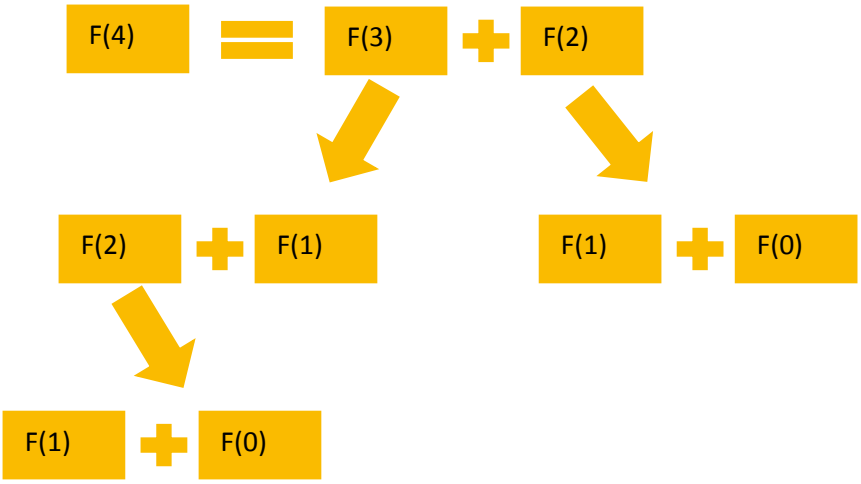
4.3. Rekurze

Rekurze je definování nějakého objektu (chápáno matematicky) pomocí sebe sama. Rekurzivní funkce musí obsahovat nějaký mechanismus, který ve vhodný moment rekurzi ukončí. Pokud by se

Rekurze je definování nějakého objektu (chápáno matematicky) pomocí sebe sama. Rekurzivní funkce musí obsahovat nějaký mechanismus, který ve vhodný moment rekurzi ukončí. Pokud by se tak nestalo, rekurze by probíhala až do „nekonečna“ Klasickým příkladem ukončení rekurze je vložení zarážek.

Klasickým příkladem využití rekurze je například výpočet Fibonacciho posloupnosti. U Fibonacciho posloupnosti je každý člen posloupnosti součtem jeho předchozích dvou prvků. A F(0) = 0 a F(1) = 1.

Příklad výpočtu čtvrtého členu je zobrazen na obrázku níže. Z Obrázku je patrno, že proto abychom mohli spočítat F(4), musíme nejdříve rekurzivně spočítat F(3) a F(2). Pro výpočet F(3) musíme nejdříve spočítat F(2) + F(1), přičemž pro výpočet F(2) musíme nejdříve rekurzivně spočítat F(1) + F(0).



Velkou část výpočtů tak rekurze dělá opakovaně, díky čemuž je výpočetně náročná. Z tohoto důvodu je často lepší využívat klasické cykly. V některých případech je celý algoritmus definován rekurzivně (například Fibonacciho posloupnost) případně nám rekurze může usnadnit práci s některými datovými strukturami.

Konkrétní příklad rekurzivního volání funkce je zobrazen níže. V našem příkladu máme 2 zarážky vyznačené žlutou barvou. Rekurzivní volání je pak vyznačeno zelenou barvou.

```
public static int fib (int n){
     if(n == 0) return 0;
     else if(n == 1) return 1;
      else return fib (n - 1) + fib (n - 2);
}
```

5. Zapouzdření

WIkipedia (ze dne 7. 5. 2018) mluví o zapouzdření:

Zapouzdření může být vysvětleno jako zabalení dat a metod do jedné komponenty. Funkce zapouzdření jsou dostupné skrze třídy ve většině objektově orientovaných programovacích jazyků. Zapouzdření rovněž umožňuje ukrytí atributů a metod v objektu pomocí stavby nepropustné zdi, která brání kód proti nechtěným změnám.

Což je velmi důležité. Pokud bychom měli složitý program. Mohli bychom omylem ovlivnit chod, nějaké jeho části. Zjištění a náprava takového problémy by nám zabralo zbytečně mnoho času. Zapouzdření je jeden ze základních konceptů objektového programování.

Zapouzdření v jazyce Java je mechanismus zabalení dat (proměnných) a kódu. V zapouzdření budou proměnné třídy skryté z jiných tříd a mohou být přístupné pouze metodami jejich současné třídy. Proto je také znám jako skrývání dat.

Dosáhneme ho tak, že části, ke kterým má mít přístup ostatní funkce označíme jako public. Tato veřejná část třídy. Je nazývána Rozhraní třídy. Do rozhraní třídy je vhodné zařadit pouze to, co o další části programu o dané třídě opravdu nutně musí vědět. U všeho ostatního, u kterého nechceme, aby mohli využívat ostatní části programu, nastavíme private.

Pokud budou nějaké části programu používat nějaké části veřejné třídy, které budou následně změněny. Může tím být ovlivněna jejich funkcionalita. Proto je lepší po zveřejnění třídy její veřejně přístupné části již neměnit.

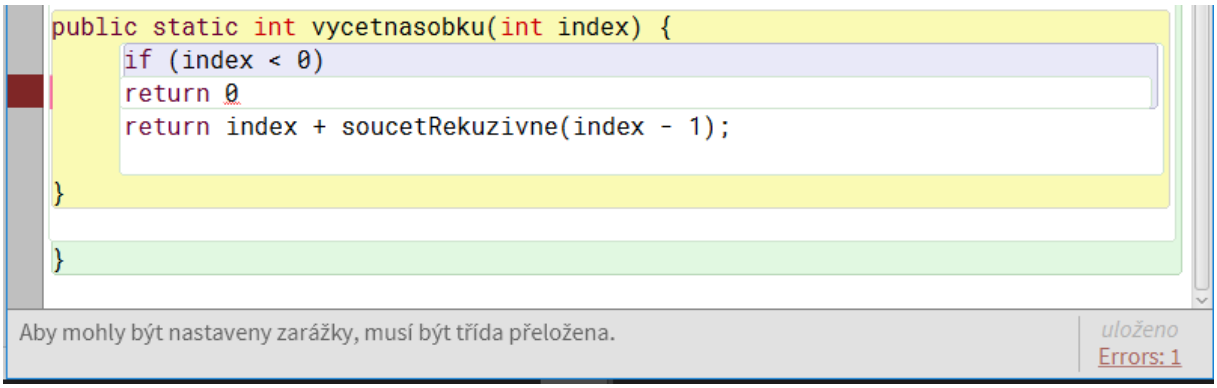
Často se deklarují všechny atributy třídy jako soukromé. Podle našeho uvážení, pak můžeme zveřejnit přístupové metody (setter, getter) díky kterým si můžeme ohlídat, například, že dotyčný atribut půjde pouze číst a ne editovat, případně ho bude možné nastavovat s nějakými omezujícími podmínkami. (Třeba věk je pouze celé kladné číslo). Shoduje-li se název atributu a názvem parametru a potřebuji-li pracovat s oběma, použiji klíčové slovo this. Toto slovo je používáno jako název třídy, ve které je metoda obsažena.

6. Ladění programu

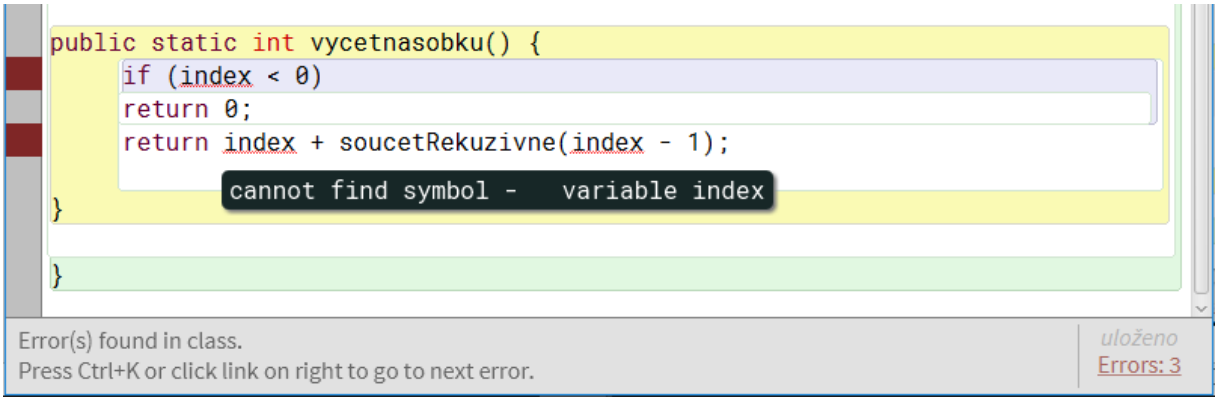
Velmi často uděláme při psaní programu nějakou chybu. Tyto chyby se dají rozdělit do několika kategorií.

Častým problémem jsou chyby syntaktické. Kdy se prohřešíme proti syntaxi jazyka. Tedy proti pravidlům jak skládat jednotlivé části. Typicky jde například o zapomenutý středník, či závorku.

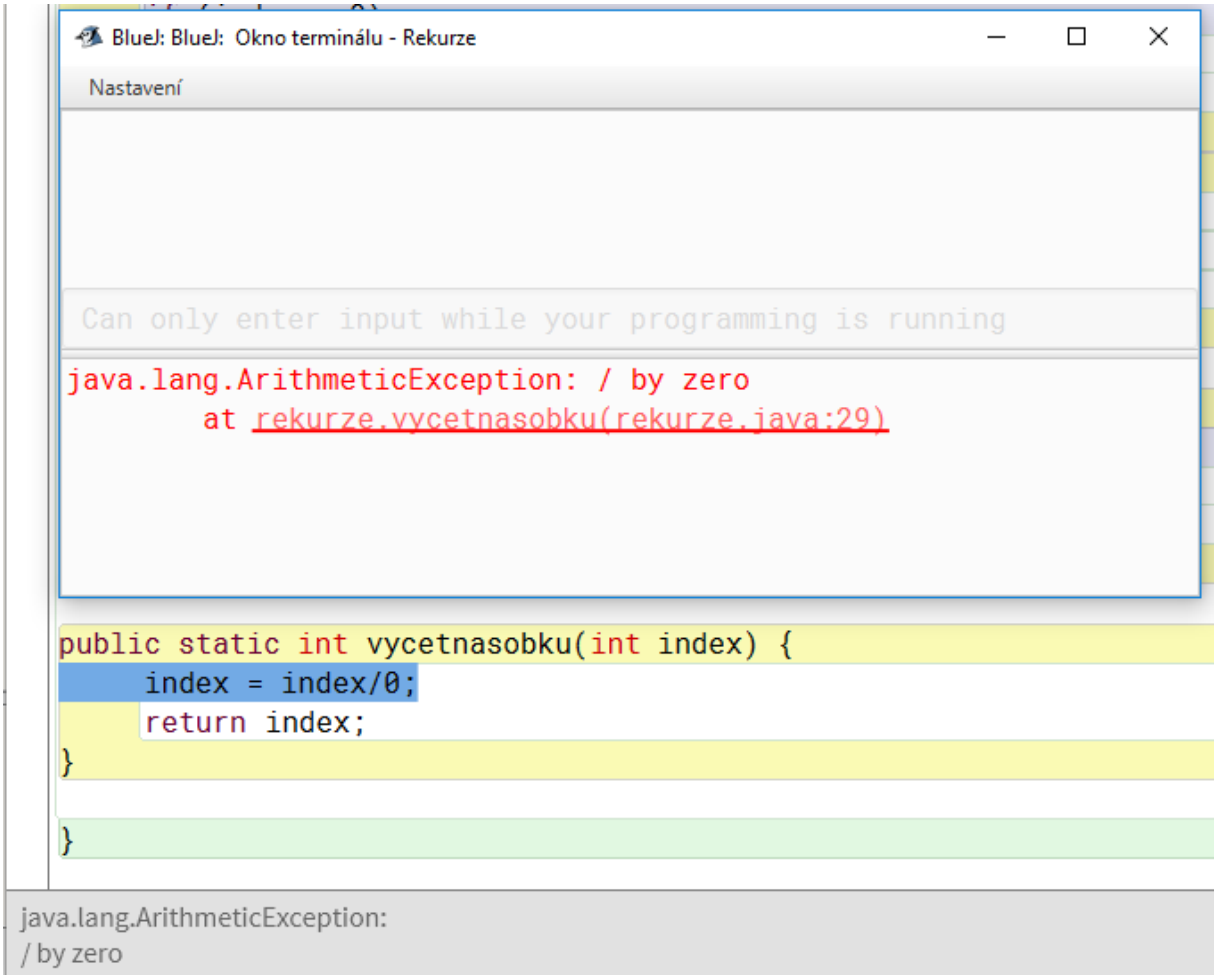
Tento problém nám překladač velmi často označí ještě před samotnou kompilací. Zároveň nám barevně vyznačí řádek, ve kterém předpokládá, problém. Jak je ukázáno na obrázku:



Další chyby se objeví při kompilaci. Pro další informace o chybě můžeme klinout vlevo dole do slova Errors. Po kliknutí se nám objeví další nápověda k chybě.



Další chyby by se dali pojmenovat, jako běhové chyby. Tyto chyby překladač nenajde. Ale v některých fázích běhu programu mohou nastat. Typickým příkladem je dělení nulou. Kde v momentě, kdy dojde k dělení nulou se program zastaví otevře se terminálové okno, ve kterém se vypíše, o jako chybu se jedná. Zároveň BlueJ označí řádek, ve kterém došlo k problému. Tak, jak je vyobrazeno na obrázku níže. Pro odstranění takového druhu chyb. Je třeba nějakým způsobem ideálně vyzkoušet všechny možné potenciálně problémové stavy programu. Tak aby se s podobnými problémy samotný uživatel nesetkal. Ideálně, když již dopředu připravíme sadu testových úloh.



Sémantické chyby jsou pak nejzáludnějším typem chyb. Kdy kompilátor nic neobjeví, program zdánlivě pracuje bez problémů. V některých neočekávaných momentech ale program vykazuje chybu. Což může být velký problém.

Příkladem může být: Havárie přistávacího modulu na Marsu (1999) Problém komunikace mezi komponentami – uživatel rozhraní očekával hodnotu v kilometrech, poskytovatel ji udával v mílích. Namísto plánovaných 140 až 150 kilometrů tedy zamířila pouze 57 kilometrů nad povrch. V té výšce je však atmosféra Marsu na sondu příliš hustá. Climate Orbiter shořel nejspíše ve výšce kolem 80 kilometrů. (Zdroj: Technet.cz)

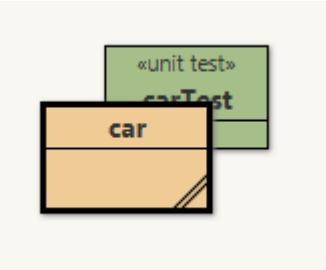
Tento typ chyb řeší softwarové inženýrství.

6.1. Programování řízené testy

Krom testování hotového programu můžeme zvolit jinou filozofii tvorby software. Test Driven Development – tedy Programování řízené testy. V rámci tohoto přístupu si nejdříve definujeme sadu testů a teprve poté píšeme samotný program, při jehož tvorbě

se pouze soustředíme na to, aby kód prošel těmito testy. (Neřešíme například efektivitu kódu) Následuje refaktorace. Odstraňují se duplicity v kódu a kód se celkově upravuje do co možná nejpřijatelnější podoby. Nové spuštění testů zajistí, že v průběhu refaktorace nedošlo k narušení funkcionality kódu.

BlueJ k tomuto účelu nabízí sadu nástrojů. V programu BlueJ vytvoříme **testovací třídu** dané třídy tak že klikneme na třídu, která má být testována pravým tlačítkem a vybereme možnost vytvořit testovací třídu. Daná třída následně dostane neoddělitelného partnera – testovací třídu. Testovací třídu barevně odliší. Jak je vidno na obrázku.



Pokud chceme testy provádět se stále stejnou sadou objektů. Můžeme si vytvořit **Testovací přípravek**. Přípravek vytvoříme tak, že si do zásobníku objektů vytvoříme pouze ty objekty, u kterých chceme, aby byly obsaženy v testovacím přípravku. Nyní klikneme pravým tlačítkem myši na testovací třídu a zvolíme možnost **Dosavadní činnost –> testovací přípravek.** Pozor do přípravku se nahrají i všechny zprávy, které jsme poslali od posledního restartu virtuálního stroje. Proto pokud chceme mít jistotu, co všechno se nahrálo, restartujeme nejdříve virtuální stroj a teprve následně tvoříme objekty. Pokud chceme přepsat již uložený přípravek jinými objekty, jednoduše tyto objekty vytvoříme a zase zvolíme **Dosavadní činnost –> testovací přípravek.** Přípravek se přepíše.

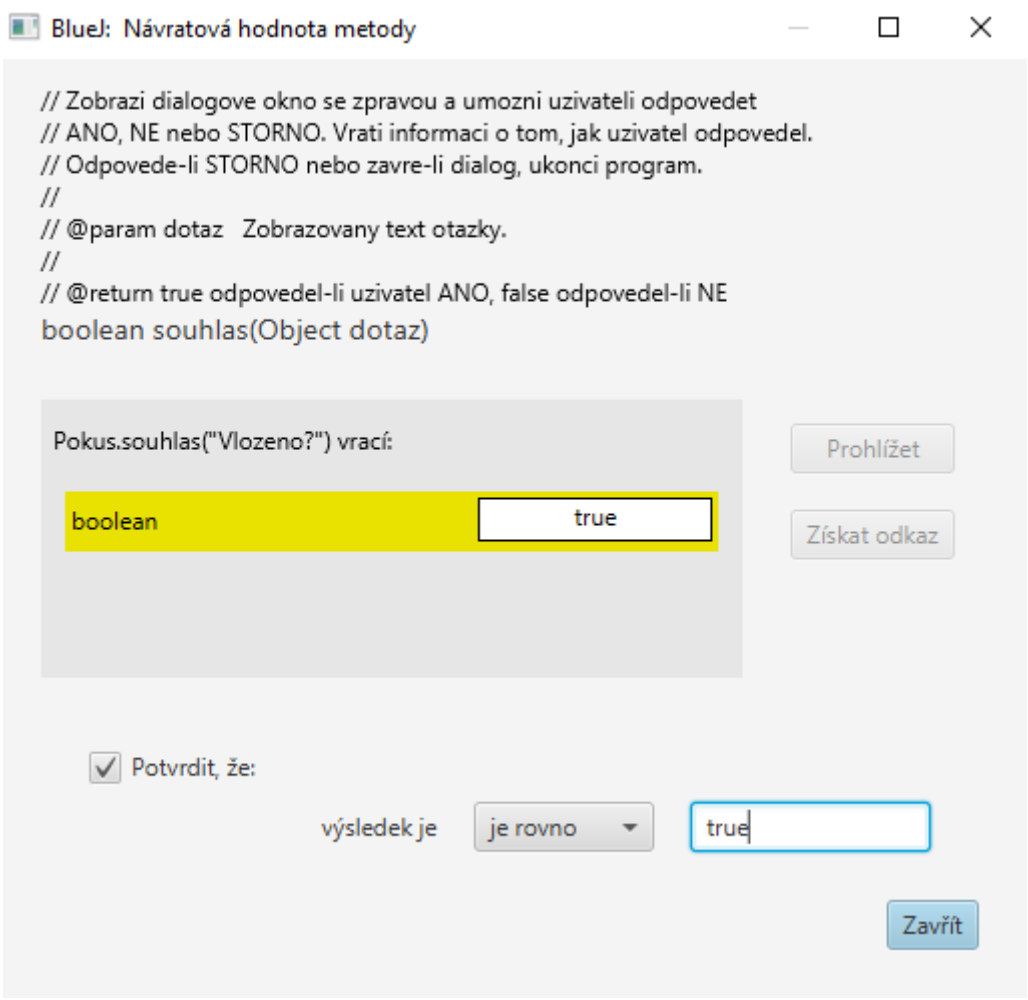
Případně můžeme ručně editovat danou testovací třídu.

Pro následné vyvolání objektů klineme pravým tlačítkem na testovací třídu a volíme **testovací přípravek –> Zásobník odkazů**.

6.2. Vytvoření testu

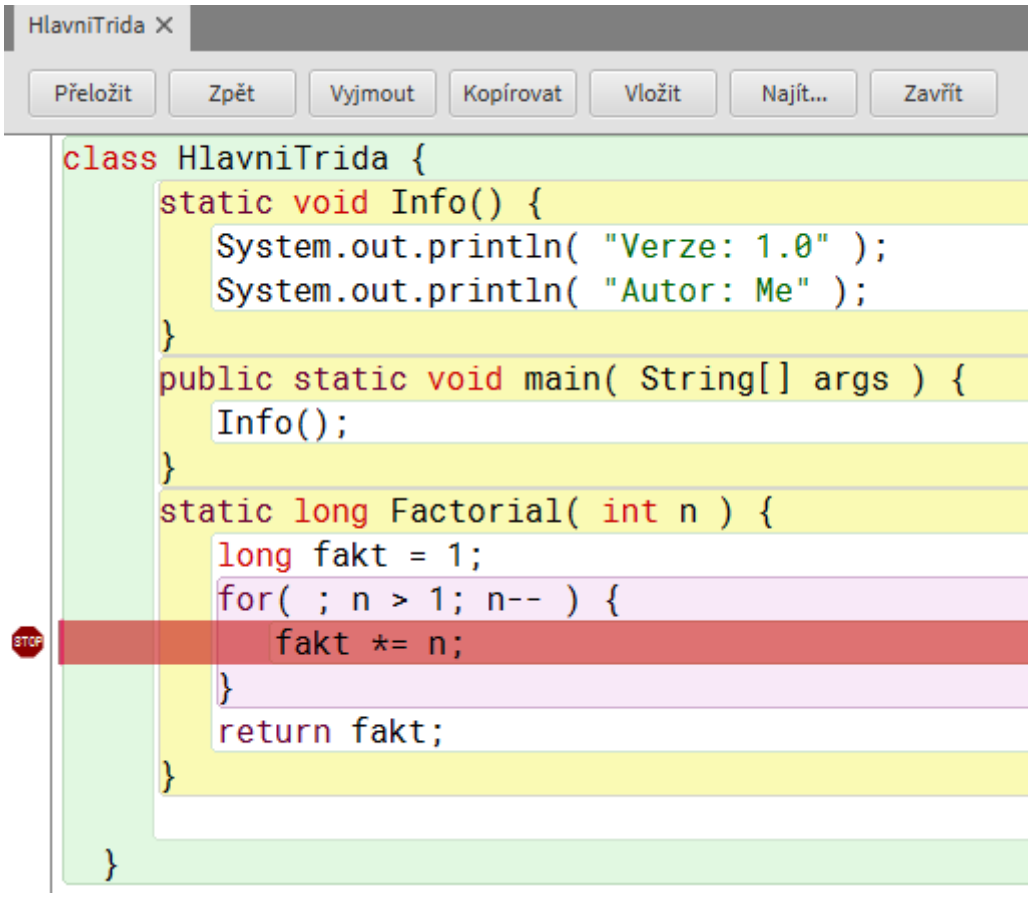
Ideálně nejdříve restartujeme virtuální stroj. Nyní klikneme pravým tlačítkem na testovací třídu a vybereme **Vytvořit testovací metodu.** Nejdříve zvolíme název samotné testovací metody**.** Nyní BlueJ nahrává náš postup, jak otestovat přípravek. Provedeme tedy požadované akce. V průběhu nahrávání se nás BlueJ bude ptát, zda u zvolených metod testovat návratovou hodnotu. Je vyobrazeno na obrázku níže. V průběhu nahrávání nám v levé části svítí červené světýlko. Pokud chceme ukončit nahrávání bez uložení, zvolíme: storno. Pro ukončení a uložení zvolíme ukončit. Obojí je hned pod červeným tlačítkem.

Pokud chceme právě vytvořený test vyzkoušet, klikneme pravým tlačítkem na testovací třídu, kde v menu vybereme test. Když v průběhu testu odpovíme jinak, než bylo nastaveno u testování návratové hodnoty. Test se přeruší. Otevře se okno s výsledky testů, kde se můžeme dozvědět, v jaké části kódu nastala chyba.


BlueJ: Návratová hodnota metody
// Zobrazi dialogove okno se zpravou a umozni uzivateli odpovedet
// ANO, NE nebo STORNO. Vrati informaci o tom, jak uzivatel odpovedel.
// Odpovede-li STORNO nebo zavre-li dialog, ukonci program.
//
// @param dotaz Zobrazovany text otazky.
//
// @return true odpovedel-li uzivatel ANO, false odpovedel-li NE
boolean souhlas(Object dotaz)
Pokus.souhlas("Vlozeno?") vrací:
boolean
true
Prohlížet
Získat odkaz
Potvrdit, že:
výsledek je
je rovno
true
Zavřít

7. Debugger

Je nástroj pomáhající k programátorovi odhalovat chyby v programu. Pro vyvolání debuggeru musíme nejdříve v kódu umístit zarážku. Což uděláme tak, že klikneme do levého sloupce v editoru kódu. Na příslušném řádku. Zobrazí se zde červená značka stop a řádek se zvýrazní červenou barvou. Jak je zobrazeno na obrázku níže. Při běhu programu se v místě zarážky běh programu zastaví a zobrazí se okno debuggeru.



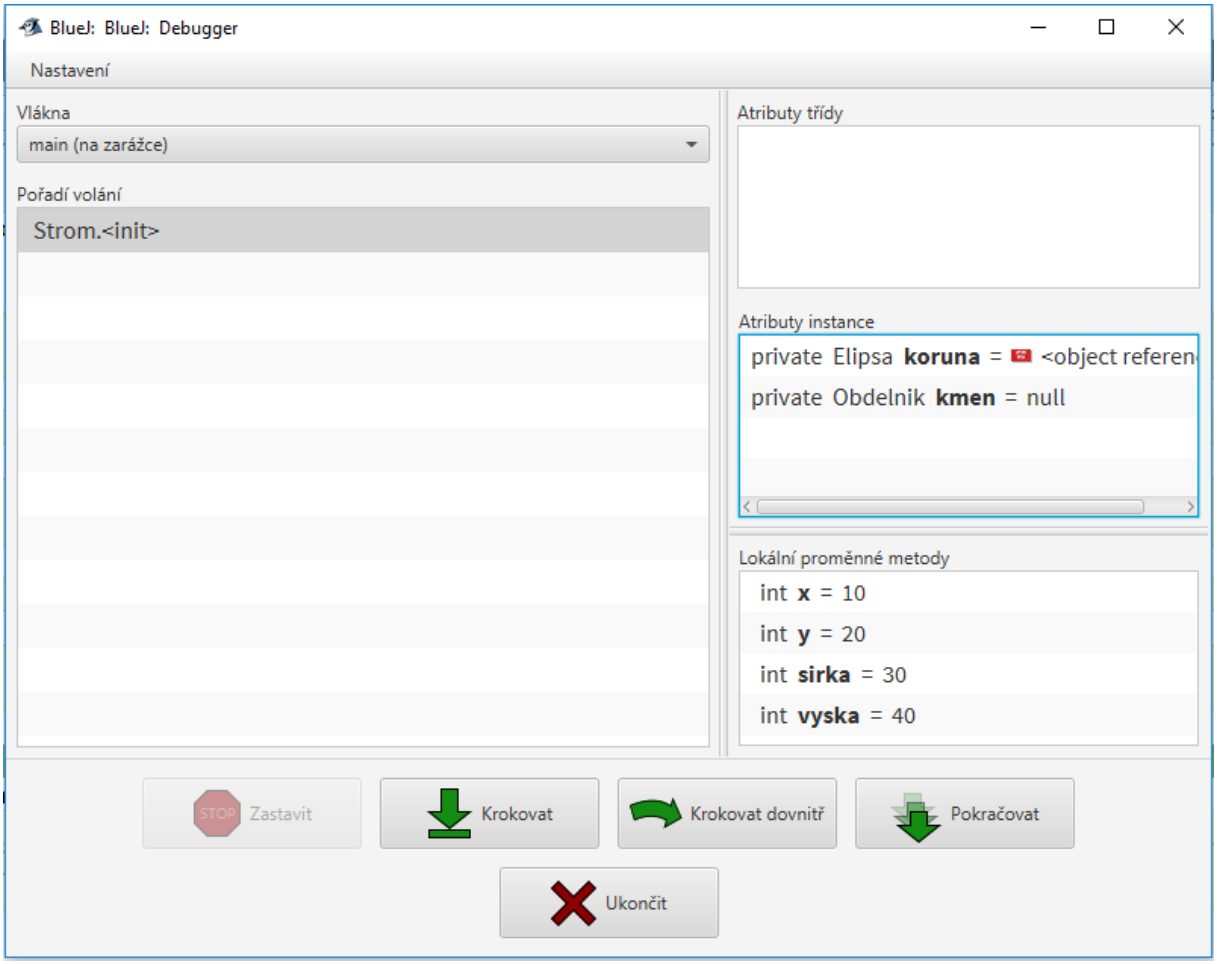
V dolní části okna debuggeru (obrázek níže) máme k dispozici celkem 5 tlačítek.

- **Zastavit** slouží k zastavení programu. Je například vhodné v případě, že se Vám zacyklí programu.
- **Krokovat** slouží k provedení dalšího kroku kódu. Příslušný krok je zároveň i graficky vyznačen zelenou barvou v editoru kódu.
- **Krokovat dovnitř** je dosti podobné, jako krokovat. Rozdíl je v tom, že v případě volání metody krokovat provede celou metodu, zatímco krokovat dovnitř, bude postupně krokovat volanou metodu.
- **Pokračovat** program bude normálně pokračovat až do svého konce, anebo do další zarážky.
- **Ukončit** ukončíme provádění programu. Případně program můžeme i ukončit

restartováním virtuálního stroje.

V průběhu krokování můžeme také přidávat to kódu nové zarážky.

Samotné okno debuggeru je rozděleno do několika částí. V levo nahoře si můžeme volit konkrétní **vlákno**. Pod volbou vlákna je oblast věnující se **Pořadí volání**. Tato oblast by se dala nazvat, jako zásobník návratových adres. Vypisují se zde všechna volání, již realizovaná při běhu programu. Nová volání se zařazují na nejvyšší pozici. Klepneme-li na konkrétní volání, otevře se editor kódu s třídou, která obsahuje volanou metodu. Barevně se zvýrazní právě vykonávaný řádek.



V pravé části okna debuggeru se vypisují **Atributy třídy**. Jde o atributy třídy, které patří zrovna vybraná složka v Pořadí volání.

Níže jsou **Atributy instance**. Zobrazují se zde atributy instance, které náleží zrovna vybrané položce v Pořadí volání. Pokud se podíváme blíže do atributů instancí v našem příkladu, zjistíme že, tato instance obsahuje dva atributy. Atribut koruna obsahuje odkaz na objekt. Pokud si na ni poklepem, zobrazí se nám okno s hodnotami všech atributů primitivních typů dané instance. Atribut kmen v daný moment ještě nemá odkaz nikam. (indikuje to hodnota null)

Pokud Ještě níže jsou zobrazeny **Lokální proměnné** také náležející právě vybrané položce v Pořadí volání. Debugger zde zobrazuje pouze proměnné, které již mají vyhrazenu paměť a přiřazenou počáteční hodnotu, tedy jsou definované.

8. Výjimky

Až do této chvíle jsme předpokládali, že v momentě, kdy dojde nějaké nepředvídané události celý program „spadne“. Tomuto problému se dá ale předejít pomocí výjimek. Při takových situacích dojde k vyvolání výjimky, která okamžitě přeruší běh programu a přechod vlákna do místa, kde je situace ošetřena. Pokud takové místo není, vlákno je ukončeno a protože zatím používáme pouze jedno vlákno, dojde k pádu celé aplikace. Existuje několik druhů výjimek, které se dají rozdělit podle příčiny, která neočekávanou situaci zavinila:

- Error – kritická chyba způsobená například nedostatkem zdrojů pro práci virtuálního stroje -*OutOfMemoryError*, přetečení zásobníku -*StackOverflowErrora* podobně. Tyto výjimky se zpravidla neošetřují.
- RuntimeException – často vznikají chybou programátora, (třeba dělení nulou -*ArithmeticException*, neplatný index - *ArrayIndexOutOf BoundsException*). Pro vyvolání této podmínky nemusíme v hlavičce metody deklarovat možnost jejich vyvolávání.
- Exception uživatele (místo telefonního čísla zadává písmena), nebo neexistující soubor, nebo soubor jiného typu. Na takovéto typy výjimek můžeme často velmi jednoduše zareagovat. Například nechat vybrat uživatele soubor ještě jednou. U takovýchto výjimek musíme vždy uvést klíčové slovo throws a seznam tříd volaných výjimek.

8.1. Chráněný režim

Potenciálně problémovou část kódu můžeme umístit do „chráněného režimu pomocí try a složených závorek. Tento režim je o trošku pomalejší. Pokud u něj ale dojde k chybě - výjimce, vykoná se část obsažená v catch. Volitelný blok finally pak bude vykonán vždy. Finally se často využívá u výjimek na uklízecí práce, kdy se vracíme do stavu před výjimkou, například zavírání souborů, uvolňování paměti a podobně.

```
public static void exception () {
      try {
            int a = 5 / 0; //create exception
      }
      catch (Exception e)
      {
            System.out.println("An exception was taken ");
      }
      finally
      {
            System.out.println("The contents of the block
            will always be processed.");
      }
}
```

8.2. Throw

Případně můžeme přidat výjimku výše pomocí klíčového slova throws. Což se týká pouze kontrolovaných výjimek, v případě že chce ošetření výjimek předat do volající metody.

Jako je uvedeno na příkladu:

```
if (index == null) {
      throw new NullPointerException();
}
```

8.3. Throws

Pokud tvoříme metodu, u které může vzniknout výjimka, kterou neumíme, anebo nechceme ošetřit. Explicitně překladači sdělíme, že tuto výjimku předáváme k ošetření do nadřazené úrovně pomocí klíčového slova throws + třídy výjimek

```
public static void  read ( ) throws IOException {
      ...
}
```

9. Práce se soubory

Veškeré informace, data, či objekty uložené v paměti se v případě vypnutí programu smažou. Pro jejich zachování a opětovné načtení je musíme uložit do nějakého souboru.

Pro bezproblémový zápis dat do soboru je vhodné ukládat data aplikace do složky appdata. Složka AppData či Application Data, případně Data aplikací obsahuje data vytvářená programy. V této složce si vytváří svou vlastní složku prakticky každý program, který je do počítače nainstalován, a do ní si následně ukládá různé údaje. Do této složky se nejjednodušeji dostanete, napíšete-li do průzkumníku souborů **%appdata%** ideální je vkládat soubory do podsložky roaming, data v této složce by měla následovat uživatele po různých počítačích v rámci domény.

Balík java.io obsahuje třídu File, která nám poskytne všechny důležité nástroje pro práci se soubory. Mnoho metod z třídy File vyžaduje jako svůj argument název souboru. Jako argument lze použít textový řetězec String, anebo instanci třídy File. Což je často optimálnější řešení, díky kterému můžeme předem o souboru zjistit nějaké informace, či s ním provést nějakou operaci.

Objekt file můžeme vytvořit několika způsoby

- Názvem souboru – vytváříme z absolutní anebo relativní cesty, která se převede na abstraktní cestu.
- Názvem souboru vzhledem k rodiči - abstraktní cesta bude vytvořena jako relativní vůči rodičovské cestě
- URI (Uniform Resource Identifier) – musí být splněny jisté požadavky. Např. cesta nesmí být prázdná

Samotný soubor lze například vytvořit takovýmto způsobem pomocí URI.

```
import java.io.File;
import java.io.IOException;
...
public void createFile() throws IOException{
      File soubor = new File("C:\\Temp\\hello.txt");
      soubor.createNewFile();
}
```

Souborový systém může implementovat omezení určitých operací na aktuálním objektu systému souborů, jako je čtení, psaní a spouštění, které nazýváme přístupová oprávnění.

Instance třídy File jsou neměnné; to znamená, že jakmile bude instance vytvořena,

abstraktní cesta, kterou představuje objekt File, se nikdy nezmění.

Třída file nabízí různé metody určené k práci se soubory. Můžeme například pracovat s:

- **Cesta k souboru** - objekt File je abstraktní cestou k souboru. S touto cestou můžeme různě pracovat. Návratové metody vracející cestu k souboru mají obvykle návratovou hodnotu typu textový řetězec Například pomocí:
    - getPath() dostaneme abstraktní cestu k souboru
    - getName() zjistíme název souboru
- **Informace o souboru** – Existuje několik metod, které nám vrací informace o souboru, jako například: length(); canRead() nebo například lastModified()
- **Informace o adresářích** – Máme k dispozici několik metod týkajících se pouze adresářů, jako například list() - vracející seznam všech souborů v adresáři, nebo unixová listRoots() Vracející pole všech kořenů adresářových stromů
- **Práce se soubory** – k samotné práci se soubory máme k dispozici různé metody:
    - renameTo(File k) jako parametr zadává další instance objektu File.
    - delete()
    - mkdir() tvorba adresáře
    - setReadOnly()

10. Grafické uživatelské rozhraní (GUI)

Je grafické prostředí, se kterým se běžný uživatel setkává a pracuje. Jednotlivé komponenty tohoto prostředí všichni velmi dobře známe. Patří mezi ně například tlačítka, roletky, posuvníky a podobně. V Javě jsou k dispozici rozdílné grafické knihovny jako například AWT (Abstract Windowing Toolkit), JFC (Java Foundation Classes). Pro BlueJ je možné nainstalovat Simple GUI Designer Extension. Které nám může zjednodušit tvorbu GUI. V tomto textu bude popsána práce s JFC, též známá jako Swing.

10.1. První aplikace

Při tvorbě aplikace s grafickým uživatelským prostředím, ve kterém aplikace poběží. Musíme nejdříve vytvořit „okno“ (rámec). Použijeme třídu JFrame. Možností jak vytvořit okno je ale vícero. Vytvořili jsme třídu KonstrukceGui, která dědí ze třídy JFrame. V této třídě jsme vytvořili bezparametrický konstruktor. Do třídy jsme vložili tlačítko a label.

Další běžně využívanou komponentou, kterou jsme ale v tomto příkladu nepoužili je pole pro zápis (JTextField). Deklarovalo by se takto:

```
JTextField someName = new JTextField(”Text”, 6)
```

V parametrech se zadává text, který se bude zobrazovat v poli. Dále je, pak číslo, které udává kolik znaků se de pole má vejít.

Rozložení komponent v okně jsme nastavili pomocí FlowLayout. Kvůli FlowLayout bylo třeba naimportovat i java.awt.

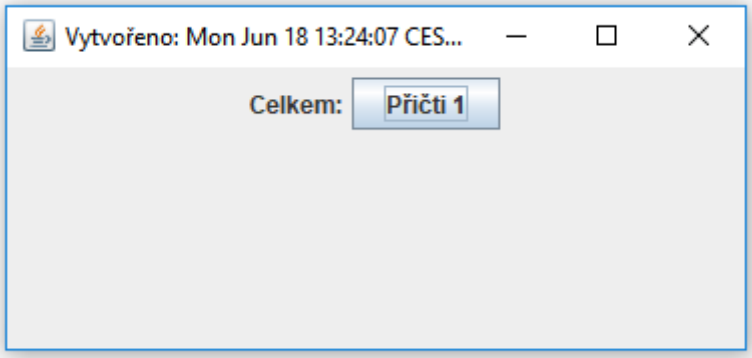
Jednotlivé komponenty bylo třeba do okna přidat. Což jsme udělali pomocí metody **add(),** u které jako parametr zadáváme název přidávaného objektu.

```
import java.awt.*;
import javax.swing.*;

public class ConstructionGui extends JFrame {
      private JButton button;
      private JLabel value;

public ConstructionGui()
{
      FlowLayout layout = new FlowLayout();
      setLayout(layout);
      value = new JLabel(”Total:”);
      add(value);
      button = new JButton(”Add 1”);
      add(button);
      }
}
```

Následně jsme vytvořili další třídu s názvem NavodGUI. S metodou main. V metodě main jsme vytvořili objekt ze třídy KonstrukceGui. Na tomto objektu jsme zavolali několik základních metod. Knihovnu date jsme importovali, abychom mohli použít funkci date(), která nám vypíše dnešní datum do titulku okna.

```
import java.awt.*;
import javax.swing.*;

public class ConstructionGui extends JFrame {
      private JButton button;
      private JLabel value;

public ConstructionGui()
{
      FlowLayout layout = new FlowLayout();
      setLayout(layout);
      value = new JLabel(”Total”);
      add(value);
      button = new JButton(”Add 1”);
      add(button);
}}
```

Samotný výsledek pak vypadá nějak takto:



Jednotlivé komponenty jsou vycentrované. Pokud budeme zmenšovat okno, budou se řadit pod sebe.

11. Události

V současnosti již máme vytvořeno jednoduché okno s jedním labelem. Po máčknutí tlačítka se ale nic nestane. Využijeme proto události, abychom přidali našemu oknu funkcionalitu. Princip událostí je takový, že v GUI vyvoláme například na máčknutí na tlačítko nějakou událost. Na objektu, kterým jsme vytvořili událost je posluchač události (event listener) Na podnět posluchače události je vyvolána metoda, která něco udělá. V našem případě vytvoříme z Labelu a tlačítka čítač máčknutí tlačítka. Kompletní kód je na další straně.

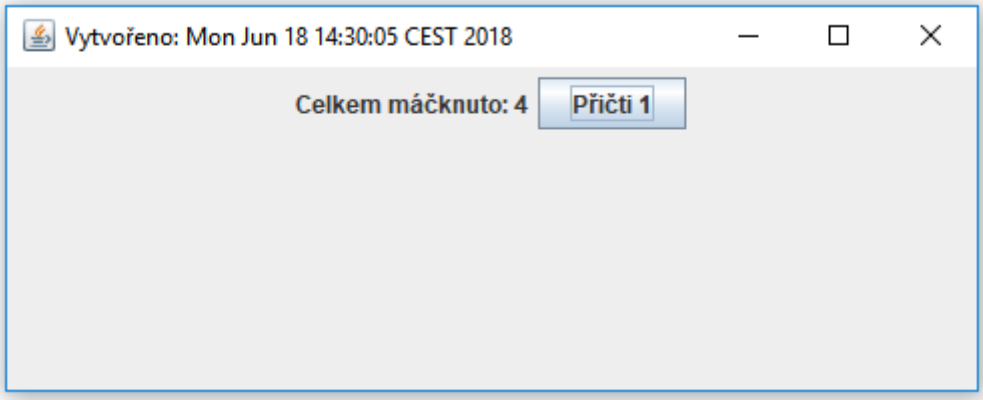
Do třídy KonstrukceGUI jsme si přidali další třídu s názvem UdálostPocet. Díky čemuž má přístup ke komponentám třídy KonstrukceGUI. Do třídy událostiPocet jsme implementovali ActionListener. Je nutné importovat java.awt.event.*;

Třída ActionListener obsahuje pouze jednu návratovou metodu actionPerformed(ActionEvent e) do které definujeme, co se má stát při vyvolání události. V našem případě jde o počítání zmačknutí tlačítka.

V konstruktoru jsme vytvořili posluchače události – objekt s názvem pocet. Tlačítku jsme přiřadili posluchače s parametrem pocet.

```
EventCost MyCount = new EventCost();
MyButton.addActionListener(MyCount);
```

Výsledek pak vypadá takto:



```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class ConstructionGui extends JFrame {
      private JButton MyButton;
      private JLabel MyValue;
      int MyNumber = 0;

      public ConstructionGui()
      {
            FlowLayout layout = new FlowLayout();
            setLayout(layout);
            MyValue = new JLabel("Not squeezed ");
            add(MyValue);
            MyButton = new JButton("Add 1");
            add(MyButton);
            EventCost MyCount = new EventCost();
            MyButton.addActionListener(MyCount);
      }

      public class EventCost implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                  MyNumber = MyNumber + 1;
                  MyValue.setText("Totally    squeezed:    "    +
            MyNumber);
            }
      }
}
```

ÚVOD DO STUDIA MÉDIÍ

1. Úvod do studia médií

Ústředním tématem předmětu je mediální komunikace jako neodmyslitelná součást moderní a postmoderní společnosti. Úvod do studia médií se zabývá základními etapami vývoje lidské komunikace se zdůrazněním role masové komunikace a jejích účinků na publikum – na příjemce mediálních sdělení. Interdisciplinární problematika je analyzována ve vztahu k její historické dimenzi.

2. Základní pojmy – médium/média, mediace, medializace

Medializace: sociální změna, jejíž podstatou je nebývalé rozšíření komunikačních médií a jejich stále zřetelnější podíl na životě společnosti.

Rysy medializace:

- **extenze**: média rozšiřují možnosti lidské komunikace
- **substituce**: média nahrazují některé sociální aktivity (televizní debata nahrazuje předvolební mítink)
- **amalgamizace**: postupně se stírají hranice mezi mediálními a nemediálními aktivitami – mediální definice reality se míchá v jeden amalgám se sociální definicí reality
- **akomodace**: média jsou významným průmyslovým odvětvím

Mediace: proces, při němž mezi dvě strany vstupuje nějaký prostředník, aby ovlivnil či zajistil vztah mezi nimi.

Médium: prostředek; prostředí; to, co zprostředkovává nějaký děj.

Médium: (v oboru mediální studia) je to důležitý článek mezi komunikátorem a adresátem například mezi redakcí novin, časopisu, rozhlasové nebo televizní redakce, internetovým (e-mailovým či facebookovým) zdrojem a čtenářem, posluchačem, divákem, účastníkem internetové diskuse apod.

Média jsou prostředky masové či mediální komunikace, které přenášejí informace v různých formách a za různým účelem, rozlišujeme:

- **tištěná média** (noviny, časopisy),
- **elektronická média** (rozhlas, televize),
- **nová média** (internet, sociální sítě -Facebook, Instagram, LinkedIn, apod.)

3. Komunikace, společnost, média, etapy ve vývoji lidské komunikace

Základní etapy ve vývoji lidské komunikace:

- epocha znamení a signálů
- epocha mluvení a jazyka
- epocha psaní
- epocha tisku
- epocha masové komunikace
- epocha počítačů a síťových médií

Marshall McLuhan (1911 – 1980) dělí lidskou komunikaci do těchto období:

- období orální kmenové kultury: doba akustického prostoru
- **období psané kultury**: akustické vnímání je nahrazeno vnímáním vizuálním
- **Gutenbergova galaxie**: tištěná kniha je prvním masovým výrobkem, prvním opakovatelným konzumním zbožím
- **Marconiho galaxie**: období spojené s nástupem elektřiny

Dnes – v období digitálních počítačových sítí můžeme mluvit o **Gatesově galaxii**, o tom však již McLuhan nepíše. **Werner Faulstich navazuje na McLuhana a určuje období lidské komunikace takto**:

- **Fáze A:** převládají **primární média**: lidé jako médium fungují do r. 1500.
- **Fáze B**: převládají **sekundární (tištěná) média**: 1500 – 1900 – tištěná média byla nejprve individuální a pak masová
- **Fáze C**: přesun těžiště na **terciární (elektronická) média**: 1900 – 2000 – převážně tisk, rozhlas, později televize
- **Fáze D**: přesun těžiště na **kvartární (digitální) média**: 2000 – dosud, trend od masovosti k individualizaci

Mark Poster vydal v roce 1995 knihu „The Second Media Age" (Druhý mediální věk), vývoj médií dělí do dvou etap:

- **První mediální věk:** vyznačuje se technologiemi šíření (broadcasting) z jednoho centra k periferiím s vysokou mírou integrace a malou mírou reciprocity (slovo „broadcasting je pro něj každé šíření typu centrum- periferie, tedy sem patří i distribuce novin a časopisů)
- **Druhý mediální věk:** je založený na vzniku komunikačních sítí, princip šíření je nahrazen principem interakce mezi jednotlivými uzlovými body sítě s malou mírou integrace a vysokou mírou reciprocity

4. Typologie sociální komunikace

Výchozí parametry pro rozlišení typů sociální komunikace:

- míra individualizace či zespolečenštění komunikačního jednání,
- míra institucionalizace,
- počet účastníků,
- míra vztahu rovnosti/nerovnosti.

Typy sociální komunikace:

- intrapersonální komunikace (se sebou samým),
- interpersonální komunikace (mezi dvěma až třemi), účastníci spolu sdílejí situační a komunikační kontext, role mluvčího a posluchačů se stírají, účastníci se vnímají jako individuality,
- skupinová komunikace (ve skupině s určitou vnitřní hierarchií – rodina, přátelé, malá pracovní skupina, do komunikace vstupuje autorita, ta ji řídí, ostatní ji respektují),
- mezi skupinová komunikace (mezi ustavenými skupinami: mezi třídami ve škole, mezi sportovními týmy, vyšší míra formalizace),
- organizační komunikace (uvnitř organizačního celku: ročník, škola, politická strana), možnosti dialogu omezeny, vztah mezi účastníky nerovný,
- celospolečenská komunikace (veškeré komunikační procesy, které jsou dostupné všem příslušníkům určité společnosti), zcela se ztrácí dialogický charakter, je primárně jednosměrná, individualizace slábne, sílí anonymita.

Celospolečenská komunikace

Rozlišují se u ní 2 zvláštní typy:

- Komunikace veřejná (přednáška, politický mítink, jednota místa a času).
- Komunikace mediální (historicky podmíněným typem mediální komunikace je MASOVÁ KOMUNIKACE).

5. Masová média, jejich znaky, funkce a vývoj

Znaky masových médií:

- sdělení jsou určena ke krátkodobému užití, mají aktuální charakter,
- masové anonymní publikum,
- veřejně přístupné informace všem bez omezení,
- převážně jednosměrný tok informací,
- odložená zpětná vazba,
- periodicita informací,
- informace nabízeny pravidelně a průběžně.

Funkce masových médií:

- informování, poskytování informací o událostech,
- socializace, vysvětlování a komentování významů událostí, podpora autorit a společenských norem, nastolování posloupnosti priorit,
- kontinuita, podpora převládajících kulturních vzorců,
- zábava, nabídka napětí a pobavení,
- získávání, agitování pro společensky významné cíle.

Vývoj masových médií

Počátek masových médií bývá spojován:

- s rozvojem technických možností produkce velkého množství výtisků stejných sdělení v relativně krátké, známé a pravidelné periodě (vydávání periodického tisku, promítání filmů, vysílání programů),
- se sociálními podmínkami pro jejich užívání (proměny publika),
- s jejich ekonomickým zhodnocením.

Periodizace vývoje masových médií

- začátek 19. století: rozvoj tzv. masového tisku,
- začátek 20. století: nástup filmu,
- 20. a 30. léta 20. století: nástup masového rozhlasového vysílání (u nás 1923),
- 50. a 60. léta 20. století: nástup masového televizního vysílání (u nás 1953),
- poslední desetiletí 20. století do současnosti: vznik a rozvoj digitálních médií.

6. Publikum

Publikum: institucionalizovaný kolektivní uživatel či příjemce nějakého sdělení.

Etapy ve vývoji publika:

- Elitní publikum: vznik čtenářské veřejnosti, publikum početně malé, vzdělané.
- Masové publikum: začíná s 1. polovině 19. století ze čtenářské obce bulvárního tiku a pokračuje až po současné televizní diváky.
- Specializované publikum: je vyvoláno rozvojem umění a vědy (specializované zájmové a vědecké časopisy, rozhlasové a televizní stanice).
- Interaktivní publikum: s nástupem nových médií (internetu a sociálních sítí) médií.

První mediální publikum byli čtenáři. Vynález knihtisku v polovině 15. století byl vynálezem záznamu a možnosti mnohočetného kopírování téhož sdělení. 4tenáři byli vystaveni velkému počtu kopií téhož sdělení. Ale čtenářská obec nebyla početná, byla omezená výše nákladů, knihy byly drahé, jejich dostupnost malá. Tato produkce nedovolovala vznik masového publika – chyběla masovost a periodicita mediální nabídky.

Se vznikem čtenářské obce vzniká i nový typ publika: část šlechty, měšťanstvo a nastupující buržoazie se začaly zajímat o své politické prosazení – začala se ustavovat kriticky diskutující veřejnost. Vznik veřejnosti se stal předpokladem pro vznik mediálního a masového publika, které potřebovalo média (tiskoviny), aby hledalo odpovědi na otázky, které si kladlo. Média se k veřejnosti obracejí jako k jednomu ze svých publik a ZÁROVEŇ veřejnost potřebuje média, aby měla kde a jak probírat témata, jež ji zajímají.

Pojetí publika

- Koncepce pasivního publika.
- Koncepce aktivního publika.
- Koncepce interaktivního publika.

7. Účinky médií a jejich fáze

- Když se uvažuje o předpokládaných účincích médií, je potřebné si uvědomit základní fakt: vliv médií na jednotlivce je sice zřejmý, ale jeho vysvětlení je obtížné a možnost důkazu není jednoznačná a reálná.
- Dále potřebné si uvědomit, že jeden a týž mediální obsah může vyvolávat u různých jedinců naprosto odlišné účinky.

Kritéria pro třídění předpokládaných účinků médií (podle Watsona, 1998):

- Co je ovlivňováno?
- V kom?
- Do jaké míry?
- V jakém časovém rozpětí?

Podle těchto otázek můžeme rozlišovat tyto účinky médií:

- **Krátkodobé a dlouhodobé:**
  - **okamžité reakce** na mediální podnět – např. zpráva o terorismu, pádu vlády, vítězství sportovce na olympiádě,
  - **dlouhodobé změny** v postojích jednotlivce nebo v uspořádání společnosti, na nichž mohou mít média svůj podíl).
- **Přímé a nepřímé:**
  - **přímé účinky** se dokazují velmi obtížně, ale můžeme je hledat například v předvolební kampani kandidáta na prezidentský úřad a jeho následným úspěchem ve volbách, k přímým účinkům lze řadit vliv reklamní kampaně na spotřebitelské chování zákazníka, pokud se konkrétní zboží po reklamní kampani skutečně úspěšně prodává,
  - **nepřímé účinky** se projevují se značným časovým odstupem a v součinnosti s dalšími faktory.
- **Plánované a neplánované**:
  - k **plánovaným** účinkům se řadí zejména: komerční, politický a sociální marketing, public relations,
  - k **neplánovaným** účinkům se řadí například násilí ve filmu odvysílaném televizí a následné agresivní chování diváka apod.).

Povaha účinků médií:

- **Kognitivní (poznávací): média nabízejí podněty, které je možné se naučit –** např. zeměpisné poznávací pořady, pořady o světě vědy a techniky, jazykové kurzy apod.
- **Citové**: zejména filmy a seriály vyvolávající dojetí, laskavost něhu, ale také třeba

strach a napětí.

- **Fyziologické**: při poslechu hudby či sledování filmu, seriálu může dojít k uvolnění, relaxaci, ale i k rozrušení a stresu.
- **Behaviorální:** jedná se zejména o změny ve spotřebitelském chování zákazníka.
- **Hodnotové** (konstruktivní či destruktivní): úcta ke slabším, znevýhodněným lidem, snaha pomoci nebo naopak snaha o násilné řešení konfliktů.

McQuail rozlišil v roce 1999 čtyři fáze výzkumu mediálních účinků:

- **Neomezená moc médií (1900 – 1940)**: přímé působení mediálních obsahů, mediální obsahy vyvolávají u příjemců shodné účinky.
- **Neúčinnost médií (1940 – 1965):**jak se odlišují jednotlivé osobnostní charakteristiky, může docházet k individualizovanému přijímání mediálních obsahů.
- **Nová víra v silné účinky médií** (1965 – 1980): aktivní přístup příjemce k médiím.
- **Transakční představy o účincích médií (od 1980 dosud):** silná pozice médií, ale současně i silná pozice publika.

8. Média a moc, propaganda

Moc (vymezení sociologické): označení vyšší pozice ve společenském vztahu (donutit někoho, aby si počínal jinak, než chtěl). Tam, kde není možnost donucení, mluví se často o vlivu.

John B. Thompson v knize „Média a modernost. Sociální teorie médií" (1995) rozlišuje **4 typy** výkonu moci ve společnosti:

- **Moc ekonomická**: vyplývá z prostředků, které vytvářejí bohatství
- **Moc politická**: vyplývá z volené nebo dosazené pozice, kde je možné přijímat rozhodnutí
- **Moc donucovací**: vyplývá z možnosti užití donucovací síly
- **Moc symbolická**: vyplývá z možnosti vytvářet a mobilizovat prostřednictvím slov, obrazů či zvuků podporu pro výkon moci

Vztah moci a médií (2 přístupy):

- Média mají natolik silnou pozici ve společnosti, že už jejich fungování lze považovat za výkon moci, kdy média fungují v rámci existujících státních útvarů, ale sama s nimi nejsou ekonomicky provázána. Pokud si mocenská elita dokáže vytvořit a prosadit nástroje na ovládání a kontrolu médií, dosáhne takové dominance nad médii, a tím i nad veřejností, že ta budou stávající moc posilovat a upevňovat.
- Ve vztahu médií a moci lze nalézt pozitivní, až ozdravující rysy. Média působí jako

hlídací pes demokracie, to znamená, že vykonávají kontrolní moc nad těmi, kdo mají moc výkonnou a legislativní (u nás prezident, vláda a parlament).

Propaganda: forma persvazivní (přesvědčovací komunikace), jde o úmyslnou manipulaci myšlenek a chování pomocí symbolů, je to plánovaná strategická komunikace. Má ofenzivní charakter, je dlouhodobá a koncepční, usiluje o formování světového názoru, o vytvoření žádoucího skupinového nebo celospolečenského vědomí a vzorů jednání.

Typy propagandy:

- politická: zaměřená na udržení a získání politické moci,
- ekonomická: zaměřená na to, aby lidé kupovali a prodávali zboží a udržovali důvěru v ekonomický systém,
- válečná (vojenská): demoralizování nepřítele nebo podpora morálky vlastního vojska a obyvatel,
- diplomatická: posílení přátelství (nepřátelství) spojenců (nepřátel),
- ideologická: šíření komplexních systémů idejí,
- didaktická: forma výchovy populace, prosazování společensky žádoucích cílů,
- eskapistická: specifická forma politické propagandy, využívá média k odvedení pozornosti od společenských problémů.

9. Typologie tištěných médií

Tištěná média: média, jejichž obsah je vázaný na papír – patří sem letáky, noviny, časopisy, knihy aj.

Kritéria typologie:

- Podle dosahu
- Podle periodicity
- Podle obsahu

Podle dosahu: tištěná média se dopravují do určitých ohraničených území a člení se na:

- **lokální** (obecní zpravodaje, vesnické noviny),
- **regionální** (Českobudějovický deník, Hlas Vysočiny),
- **nadregionální** (Šumavský zpravodaj),
- **celostátní** (Lidové noviny, Hospodářské noviny),
- **nadnárodní** (Reader´s Digest).

Podle periodicity: periodicitu definuje zákon o právech a povinnostech při vydávání periodického tisku. Periodický tisk: jsou noviny, časopisy a jiné tiskoviny vydávané pod stejným názvem, se stejným obsahovým zaměřením a v jednotné grafické úpravě nejméně 2krát v kalendářním roce.

- **Noviny:** tištěná média (periodika), která vycházejí minimálně dvakrát týdně a obsahují aktuální politickou část, která se vyznačuje tematickou pestrostí (diverzitou).
- **Časopis:** tištěné médium, vychází v delších intervalech než noviny, maximálně jedenkrát týdně a minimálně dvakrát ročně.
- Za periodická tištěná média se NEPOVAŽUJÍ: Sbírky zákonů a úřední věstníky.

Podle obsahu:

- zpravodajské týdeníky (Týden, Instinkt),
- společenský a životní styl (Květy, Vlasta, Xantypa),
- pro děti a mládež (ABC, Bravo, Dívka),
- zájmy a hobby (Golf, Tenis, Cyklistika, Včelařství).

10. Čtyři teorie tisku

Čtyři teorie tisku formulovali v roce 1956 v článku „Four Theories of the Press“ mediální analytici Fredrich Siebert, Theodor Peterson, Wilbur Schramm.

Jedná se o čtyři přístupy řešení vztahu mezi společností (politickým režimem) a médii:

- **Autoritářská teorie**: média jsou prostředek pro sdělování postojů a názorů nějaké autority (například panovník nebo politik, který je vlastníkem médií) a souhlasí se stávajícím rozdělením moci.
- **Libertariánská teorie**: je označována také jako teorie svobodného tisku. Zabraňuje, aby stát kontroloval veřejnou komunikaci - všechny mediálně prezentované názory jsou v rovnováze. Média jsou uspořádána tak, aby mohla kdykoli říci, co se jim zachce.
- **Teorie společenské odpovědnosti**: média by měla nacházet rozličné pohledy na daný problém, ale měla by mít hranici, za niž nejdou (například nepodporovat násilí, zločinnost, terorismus atd.).
- **Sovětská komunistická teorie médií**: média jsou nástroj jednoho typu socializace a formování veřejného mínění, média slouží jako prostředek vzdělávání a osvěty (výchova socialistického občana).

Denis McQuail navrhl další 2 koncepce:

- **Rozvojová teorie médií**: média by měla přispívat k modernizaci společnosti, měla by naplňovat socializační a vzdělávací cíle.
- **Teorie demokratické participace**: média představují společenskou instituci, nemá existovat žádná demokratická kontrola médií, média by měla být uspořádána tak, aby podporovala zájmy menšin a jednotlivců.

11. Vysílání veřejné služby

Mezi média s vysílaným signálem se řadí televizní a rozhlasové organizace:

- veřejného sektoru (státní, veřejnoprávní – vysílání veřejné služby),
- soukromého sektoru.

Veřejný sektor v oblasti vysílacích médií se liší stupněm svázanosti se státním aparátem a vládou:

- **Státní rozhlas a státní televize** jsou bezprostředně podřízeny státnímu aparátu – v zemích s autoritářským režimem mu jednostranně a bezvýhradně slouží.
- **Veřejnoprávní rozhlas a televize** jsou zřizovány zákonem s tím, že jsou nestátní organizací, vykonávající neziskovou veřejnou službu.

Soukromý sektor v oblasti vysílacích médií je organizován na bázi soukromého podnikání, poprvé se objevilo a dosud existuje hlavně v USA. V Evropě rozhlasové vysílání začínalo jako licencované soukromé podnikání, jakmile se ale začal rozhlas masové rozšiřovat, národní státy v Evropě jeho vysílání zestátnily. Například do společnosti Radiojournal u nás vstoupil v roce 1925 většinovým podílem československý stát. Také soukromá BBC (British Broadcasting Company) se v roce 1927 stala veřejnou korporací BBC (British Broadcasting Corporation). !!! BBC se stala průkopníkem: na rozdíl od ostatních evropských rozhlasů se nestala součástí státního aparátu, ale zůstala veřejnou korporací nezávislou na státu.

Veřejnoprávní modely vysílání se v západoevropských demokraciích rozvíjely po 2. světové válce, až po roce 1989 ve východní Evropě. Veřejnoprávní rozhlasy a televize zůstávaly nadále monopolním vysílatelem v dané zemi – limity kmitočtového spektra bránily vzniku soukromého vysílání, roli hrála i investiční a provozní náročnost vysílání.

Rozvoj soukromého rozhlasového vysílání v západní Evropě 70. léta 20.století, rozvoj soukromého televizního vysílání v západní Evropě 80. léta 20. století. – To znamená, že v této době vzniká **duální systém vysílání – současně existuje veřejnoprávní a soukromé rozhlasové a televizní vysílání.**

12. Mediální výchova a mediální gramotnost

Mediální gramotnost: znalosti a dovednosti, které umožňují chápat a kriticky vyhodnocovat různé aspekty médií (mediálních obsahů)

Mediální výchova: systematické předávání poznatků o médiích.

Roviny mediální gramotnosti:

- **Mediální výchova**: - **školní** (viz průřezová témata Rámcového vzdělávacího programu pro základní školy a střední školy); - **mimoškolní** (zájmové kroužky mladých novinářů)
- **Profesní vzdělávání**: - vzdělávání učitelů, i budoucích učitelů, kteří učí nebo budou učit mediální výchovu; vzdělávání novinářů
- **Mediální osvěta**: - **mediální kritika**: pořady o médiích v rozhlase a v televizi; označování vhodnosti pořadů v televizi (např. pořad není vhodný pro děti)

13. Významné osobnosti českých masových médií

Josef Kajetán Tyl (1808 – 1856)

- spisovatel, dramatik, novinář,
- 1833 založil časopis *Jindy a nyní* – ten přejmenoval na *Květy české* - chtěl vytvořit český časopis, který by nekopíroval zahraniční vzory,
- časopis s názvem *Květy* vychází dodnes,
- snaha o osvětu a veřejnou aktivizaci zejména venkovského publika.

Karel Havlíček Borovský (1821 – 1856)

- 1846 redaktor Pražské noviny,
- 1848 založil první český deník *Národní noviny,,*
- 1849 založil časopis *Slovan* (*Národní noviny* byly z politických důvodů zastaveny), i zde veřejně kritizuje rakouské politické poměry,
- 1851 exil Brixen (Švýcarsko),
- 1855 zpět v Čechách,
- zastánce austroslavismu: spolupráce slovanských národů v rámci habsburské monarchie,
- nekompromisní kritické myšlení, logické argumenty, kultivovaný český jazyk.

Julius Grégr (1831 – 1896)

- český politik a novinář,
- 1862 majitelem a redaktorem novin *Národní listy*, zde se mu podařilo soustředit tehdejší výkvět českého novinářství – v *Národních listech* publikovali například Jakub Arbes a Jan Neruda.

František Gel (1901 - 1972)

- 1924 – Lidové noviny – redaktor,
- 1945 redaktor politického vysílání Československého rozhlasu, zpravodaj z Norimberského procesu,
- 1955 učitel žurnalistiky na Filosofické fakultě Univerzity Karlovy.

Pavel Tigrid (1917 – 2003)

- spisovatel, publicista, politik,
- představil českého protikomunistického exilu,
- 1939 emigrace do Anglie, zde se v Londýně podílí na exilovém vysílání BBC,
- 1945 návrat do Prahy, šéfredaktor časopisu *Obzory,*

- 1948. opět emigrace, redaktor rozhlasového vysílání Svobodná Evropa, redaktor časopisu *Svědectví,*
- po roce 1989 opět návrat do Československa, poradce prezidenta Václava Havla, ministr kultury.

14. Literatura

JIRÁK, J., KÖPPLOVÁ, B. 2009. Masová média. Praha: Portál, ISBN 978-80-7367-466-3

JIRÁK, J., KÖPPLOVÁ, B. 2003. Média a společnost. Praha: Portál, ISBN 80-7178-697-7

KONČELÍK, J. a kol. 2010. Dějiny českých médií v datech. Praha: Portál, ISBN 978-80-7376-698-8

OSVALDOVÁ, B., HALADA, J. 2007. Praktická encyklopedie žurnalistiky a marketingové komunikace. Praha: Libri, ISBN 978-80-7277-266-7

REIFOVÁ, I. a kol. Slovník mediální komunikace. Praha: Portál, 2004, ISBN 80-7178-926-7

SCHULZ, W. a kol. 1998. Analýza obsahu mediálních sdělení. Praha: Karolinum, ISBN80-7184-548-5

IT SECURITY

1. Motivace

- Silná závislost moderní společnosti na informační a komunikační technologii (IKT, zejm. kritická infrastruktura).

- IKT se stává kritickou informační infrastrukturou (Critical Information Infrastructure, CII).

- následující trendy činí ochranu IKT ústředním tématem (mj. podle [Eckert 2008]):
  - globalizace
  - mobilita (smartphone, informace „na dosah ruky“)
  - propojení
  - Všudypřítomný computing / všudypřítomné výpočty, Internet věcí (Internet of Things - IoT)
  - Průmysl x.0
  - Smart Home/Smart Grid (Inteligentní sítě)/Car/*
  - Cyber War (Kybernetická válka)/Information Warfare (informační válka)/Espionage(Špionáž)/*

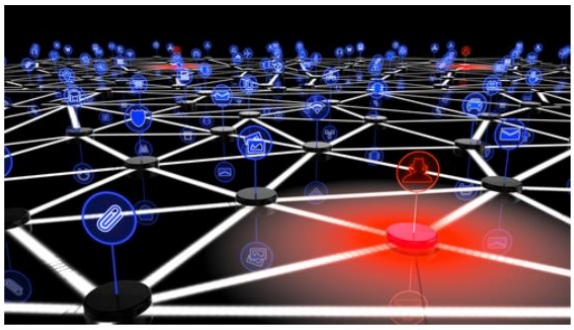
- Závěr: Ochrana IKT má pro společnost klíčový význam!

KrebsonSecurity
In-depth security news and investigation

21 KrebsOnSecurity Hit With Record DDoS

SEP 16

On Tuesday evening, KrebsOnSecurity.com was the target of an extremely large and unusual distributed denial-of-service (DDoS) attack designed to knock the site offline. The attack did not succeed thanks to the hard work of the engineers at **Akamai**, the company that protects my site from such digital sieges. But according to Akamai, it was nearly double the size of the largest attack they'd seen previously, and was among the biggest assaults the Internet has ever witnessed.



The attack began around 8 p.m. ET on Sept. 20, and initial reports put it at approximately 665 Gigabits of traffic per second. Additional analysis on the attack traffic suggests the assault was closer to 620 Gbps in size, but in any case this is many orders of magnitude more traffic than is typically needed to knock most sites offline.

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUM

RISK ASSESSMENT —

Why the silencing of KrebsOnSecurity opens a troubling chapter for the 'Net

"Free speech in the age of the Internet is not really free," journalist warns.

DAN GOODIN - 9/23/2016, 10:58 PM

DDoS-Attacke

Mit Todessternen auf Spatzen schießen

Die Website eines Journalisten wurde von einem Botnetz aus Haushaltsgeräten bombardiert. Die bisher wohl größte DDoS-Attacke überhaupt war aber nur ein Vorgeschmack.

Von **Patrick Beuth** ZEIT ONLINE

26. September 2016, 15:18 Uhr / 46 Kommentare

The banker that can steal anything

By Anton Kivva on September 20, 2016. 10:58 am SECURELIST

MOBILE

BANKING TROJAN GOOGLE ANDROID MOBILE BROWSER MOBILE MALWARE

CONTENTS

Anton Kivva

In the past, we've seen superuser rights exploit advertising applications such as Leech, Guerrilla, Ztorg. This use of root privileges is not typical, however, for banking malware attacks, because money can be stolen in numerous other ways that don't require exclusive rights. However, in early February 2016, Kaspersky Lab discovered Trojan-Banker.AndroidOS.Tordow.a, whose creators decided that root privileges would come in handy. We had been watching the development of this malicious program closely and found that Tordow's capabilities had significantly exceeded the functionality of most other banking malware, and this allowed cybercriminals to carry out new types of attacks.

Technology

Dropbox hackers stole 68 million passwords - check if you're affected and how to protect yourself

share The Telegraph

The details of tens of millions of Dropbox users are for sale online CREDIT: DROPBOX

By Cara McGoogan

31 AUGUST 2016 • 11:05AM

heise Security

« Vorige | Nächste »

Android-Schädling Tordow: Banking-Trojaner mutiert zum Super-Trojaner

28.09.2016 14:00 Uhr – Dennis Schirrmacher

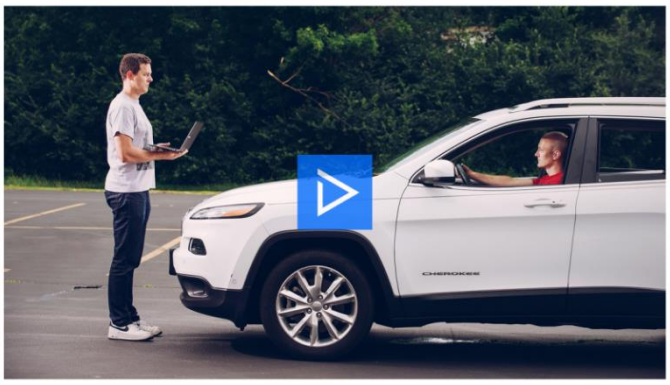
vorlesen

(Bild: Christoph Scholz, CC BY-SA 2.0)

Sicherheitsforscher warnen vor einer Banking-Malware, deren Funktionsumfang sämtliche Träume von Kriminellen erfüllen dürfte: Der Trojaner kann im Grunde alles mit infizierten Android-Geräten anstellen.

HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY—WITH ME IN IT

WIRED



43 million passwords hacked in Last.fm breach

Posted Sep 1, 2016 by John Mannes (@JohnMannes)

Crikey: 43,570,999 user accounts were breached in a hack of Last.fm that occurred in March of 2012, according to a report from LeakedSource. Three months after the breach, in June of 2012, Last.fm issued the following statement:

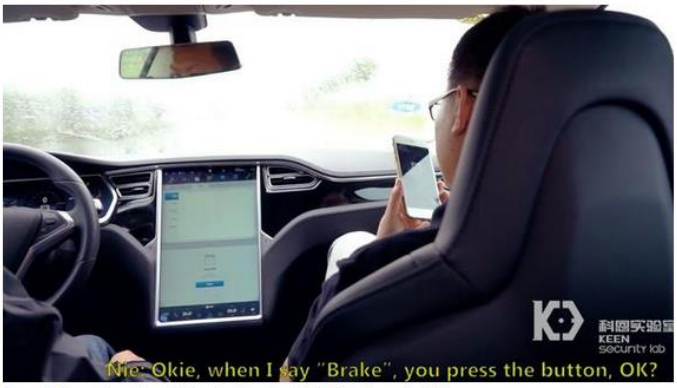
heise Security

« Vorige | Nächste »

Tesla Model S lässt sich von fern kapern

20.09.2016 08:24 Uhr – Andreas Wilkens

vorlesen



Screenshot aus dem Demo-Video der Forscher (Bild: Keen Security Lab)

Forscher des chinesischen Internetunternehmens Tencent demonstrieren, wie sie manche Funktionen eines Tesla Model S unautorisiert von fern steuern. Dabei gelang es ihnen auch, ein fahrendes Auto anzuhalten.

derStandard.at

Bis zu fünf Millionen Betroffene bei Facebook-Hack aus EU

2. Oktober 2018, 07:48

3 POSTINGS

foto: dado ruvic / reuters

Der Facebook-Hack sorgt weiter für Aufregung.

Weniger als zehn Prozent der 50 Millionen weltweit – Unternehmen verspricht "bald" neue Informationen

Von den fast 50 Millionen von einem Hacker-Angriff betroffenen Facebook-Nutzer stammen weniger als zehn Prozent, das sind maximal 5 Millionen, aus der Europäischen Union. Das teilte die zuständige irische Datenschutzbehörde am Montagabend bei Twitter mit. Facebook habe zugesichert, "bald" ausführlichere Informationen liefern zu können, hieß es in der knappen Stellungnahme weiter.

Car hacking is the future - and sooner or later you'll be hit

theguardian

Security is finally being taken seriously but the fact that we are increasingly entrusting our lives to self-driving cars creates unease

A Tesla self-driving car on autopilot mode. Researchers explored the potential ways in which such vehicles could be hacked or exploited. Photograph: Bloomberg via Getty Images

DSGVO: Mehr als 1000 US-Portale für Europäer gesperrt

futurezone

08.08.2018

© Bild: Maksim Kabakou - Fotolia / Maksim Kabakou/Fotolia

Noch immer sind mehr als 1000 US-Nachrichtenseiten hierzulande nicht erreichbar, Instapaper ist jedoch wieder verfügbar.

DSG: Verwaltungsstrafe bis EUR 25.000,-- DSGVO: Geldbußen bis EUR 20.000.000,--

https://www.dataprotect.at/info/geldbußen/österreich/

Auch bereits bei **Geltung des Datenschutzgesetzes** (bis 24.5.2018) gibt es **Geldstrafen für Datenschutzverletzungen**; der Strafrahmen ist jedoch (relativ) gering und reicht bis zu **EUR 10.000,--** (für geringe Verstöße) oder bis zu **EUR 25.000,--** (für gröbere Verstöße); es gibt auch **gerichtlich strafbare Handlungen** im DSG (§ 51: Datenverwendung in Gewinn- oder Schädigungsabsicht).
Der **Strafrahmen für Datenschutzverletzungen** steigt mit Geltung der **DSGVO** (25.5.2018) **dramatisch**, und zwar auf 4 % des weltweiten Konzernumsatzes des Vorjahres / EUR 20.000.000,-- als maximale Strafe bzw. 2 % des weltweiten Konzernumsatzes des Vorjahres / EUR 10.000.000,-- bei "geringfügigeren Delikten", wobei dies jeweils die absolute Obergrenze darstellt und immer der Betrag anwendbar ist, der "höher" ist.

dataprotect it-recht

ENTSCHEIDUNGEN ZUM DATENSCHUTZRECHT · 20. September 2018

erste Geldstrafe durch die DSB in Österreich

Die DSB hat die erste **Geldstrafe** verhängt.

EUR 4.800,-- für eine nicht korrekt gekennzeichnete Videoüberwachung, die den öffentlichen Raum mitüberwachte.

https://www.dataprotect.at/2018/09/20/erste-geldstrafe-durch-die-dsb-in-österreich/

Nach Medienberichten (Salzburger Nachrichten, 19.09.2018) wurde gegen einen Betreiber eines Wettlokals in der Steiermark eine Geldstrafe von EUR 4.800,-- verhängt, weil eine **Videoüberwachung nicht ausreichend gekennzeichnet** war und ein **großer Teil des Gehsteigs von der Anlage mitaufgezeichnet** wurde. Die Überwachung des öffentlichen Raums in dieser Art und Weise, nämlich großflächig durch Private ist nicht zulässig.

2.Cíle ochrany informační bezpečnosti

Úkolem informační bezpečnosti (popř. IT bezpečnosti) je zajištění těchto cílů ochrany (základních hodnot) [Stallings 2006]:

1. Důvěrnost (angl. confidentiality)
2. Integrita (angl. integrity)
3. Dostupnost (angl. availability)

C-I-A

4. Autentizace (angl. authenticity)
5. Nepopiratelnost (angl. accountability nebo non-repudiation)

2.1. Důvěrnost

- **důvěrnost** = angl. confidentiality
- *Ochrana dat před neoprávněným zveřejněním.* [Stallings 2006]
- **Traffic Flow Confidentiality** = ochrana před analýzou toku dat
- důvěrnost v běžném životě
  - listovní tajemství & telekomunikační tajemství
  - služební tajemství
  - zpovědní tajemství
  - povinnost mlčenlivosti právních zástupců/notářů/lékařů
  - dohody o zachování mlčenlivosti/NDAs (non-disclosure agreements)
- dosažitelné prostřednictvím zakódování (šifrování)
  - symetrické šifrování
    - společně sdílený klíč (stejný klíč pro za- i odkódování) => problém s distribucí klíče; proudová/bloková šifra, substituce & transpozice
  - asymetrické šifrování
    - veřejný a soukromý klíč (veřejný klíč pro zašifrování a soukromý pro odšifrování) => problém s distribucí klíče vyřešen, algoritmicky náročnější, a proto pomalejší

- v praxi hybridní postup: předání generovaného klíče relace prostřednictvím asymetrické kryptografie a poté symetrického šifrování

2.2. Integrita

- **integrita** = angl. integrity
- *Ujištění, že přijatá data jsou přesně taková, jaká byla zaslána pověřeným subjektem (tzn. nebyla změněna, nebyla do nich vložena další data, nebyla smazána ani opakovaně přehrána)* [Stallings 2006]
- Změnám (způsobeným např. chybou v přenosu nebo aktivními útoky) **není** možné **zabránit**, avšak je možné ji **rozpoznat**!
- Autentizace a integrita jsou často považovány za jeden celek. Jedno bez druhého je nepoužitelné.
- dosažitelná použitím (**kryptografických**) **kontrolním součtem**, např.
  - CRC (Cyclic Redundancy Check – Cyklický redundantní součet) → pouze pro chybu v přenosu
  - (Keyed-Hash) Message Authentication Code (MAC, HMAC)
  - elektronické podpisy

Kryptologická hašovací funkce nebo **kryptografická hašovací funkce** je zvláštní forma hašovací funkce, která je odolná vůči kolizím nebo je jednosměrnou funkcí (nebo obojí).

Hašovací funkce je funkcí, která zobrazuje posloupnost znaků libovolné délky na posloupnost znaků s pevnou délkou. Matematicky není tato funkce injektivní (zleva jednoznačná) a bezpodmínečně surjektivní (napravo úplná).

2.3. Dostupnost

- **dostupnost** = angl. Availability

- *Dostupnost je vlastnost systému nebo systémového zdroje být přístupný a použitelný ba vyžádání oprávněnou systémovou entitou dle specifikace výkonu systému.* [Stallings 2006]
- Typy výkonu
  - využitelný, popř. aplikovatelný
  - dostatečná kapacita
  - jasný pokrok a/nebo jasně definované čekací doby
  - dokončení v přijatelném časovém rámci
  - ...

- Přístupy k realizaci/záruka dostupnosti
  - technický: tolerance chyb a redundance prostřednictvím např. RAID, Clustering, ..., zrcadlová datová centra, nepřerušované dodávky el. proudu, ...
  - organizační: Service Level Agreements (SLAs – smlouva o službě), klasifikace dostupnosti

- Klasifikace dostupnosti
  - 2-6: 99% (výpadek ~ na 90 hodin ročně)
  - 99.9%, ... do 99.9999% (= výpadek ~ na 30 sekund ročně!).

2.4. Autentizace

- **autentizace** = angl. Authenticity
  - autentizace entity= Vím, s kým komunikuji
  - autentizace dat= Vím, od koho data přicházejí

- autentifikace a autentikace (authentication)

- Jistota, že komunikující entita je tou, za kterou se vydává. [Stallings 2006]

- znaky autentizace = znak, jímž ověřena identita účastníka
  - na základě znalostí (PIN, heslo)
  - na základě vlastnictví (klíč, karta)
  - na základě vlastnosti (biometrické znaky jako otisk prstu, duhovka, hlas, podpis, sítnice, ...)
    - Sítnice = oční pozadí (sken prostřednictvím infračervených laserových paprsků)

        - Sken duhovky (prostřednictvím normální optické kamery)

- kontrola přístupu (Access Control)
    - autentizace je předpokladem pro kontrolu přístupu!

2.5. Kontrola přístupu

- **Kontrola přístupu** = angl. access control

- Prevence neoprávněného použití zdrojů. [Stallings 2006]

- Autentizace přistupující entity musí být zajištěna.

- Ověřuje se,
    - kdo smí mít přístup ke zdroji,
    - za jakých podmínek smí přístup proběhnout a
    - jaká práva smí přistupující entita mít.

- Princip nutných znalostí (Need-to-know principle)

- základní modely
    - Discretionary Access Control (DAC)
        - stanovení práv výhradně na základě identity uživatele
    - Mandatory Access Control (MAC)
        - nejen identita uživatele, nýbrž i dodatečná pravidla a vlastnosti
    - Role Based Access Control (RBAC)
        - přidělení práva na základě role, tj. nejen na základě identity uživatele, nýbrž i na základě role uživatele (příslušnost ke skupině)

2.6. Nepopiratelnost

- **nepopiratelnost** = angl. accountability nebo non-repudiation
- Poskytuje ochranu před tím, aby žádná entita účastnící se komunikace nemohla popřít část nebo celou účast na této komunikaci. [Stallings 2006]
- nepopiratelnost odesílatele = odesílatel nemůže odeslání zprávy popřít.
- Nepopiratelnost doručení = příjemce nemůže přijetí zprávy popřít.
- dosažitelné prostřednictvím elektronických podpisů (nikoli prostřednictvím [H]MACs!).

3. Informační vs. IT bezpečnost

- Informační bezpečnost (Information Security, InfoSec) se zabývá daty v libovolné podobě (elektronická, písemná, ústní).
- IT bezpečnost (IT Security, IT-Sec) se soustředí výhradně elektronické zpracování dat.
- Definice IT bezpečnosti (přenositelná na InfoSec, dle [BSI 1992]):

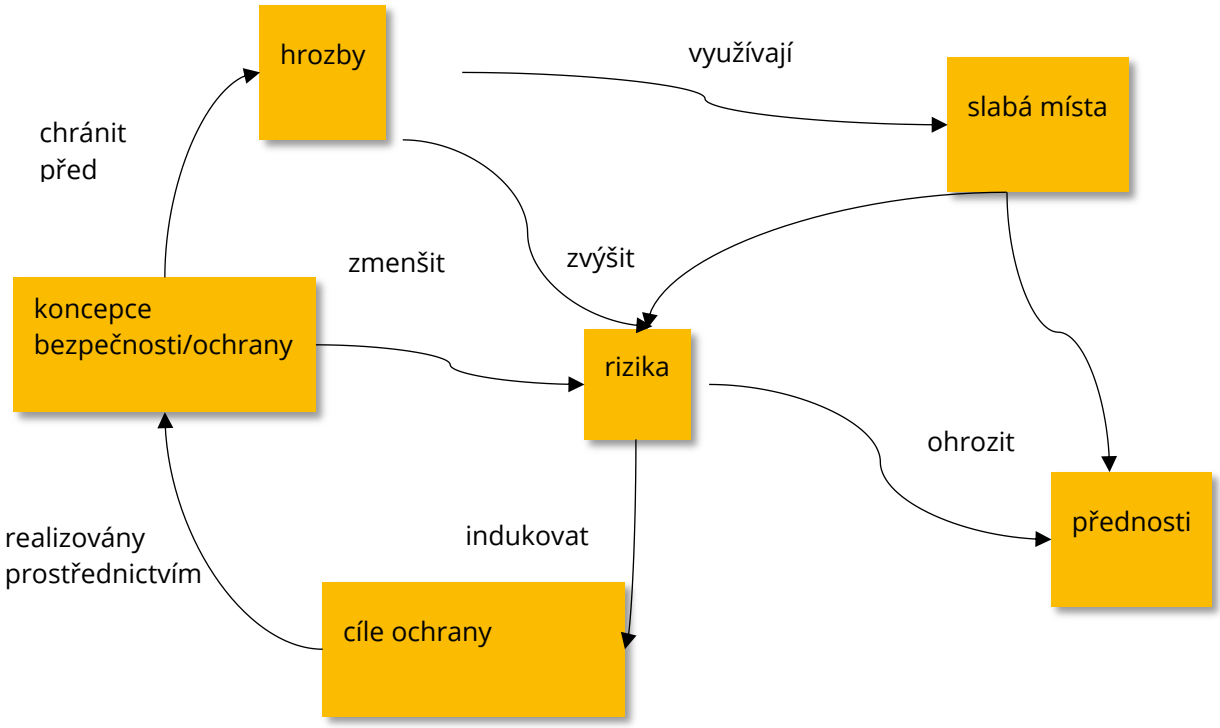
> IT bezpečnost je stav IT systému, v němž jsou **rizika**, která se vyskytují při použití tohoto IT systému na základě **hrozeb**, prostřednictvím **přiměřených opatření** omezena na **únosnou (přijatelnou) míru**.

- Nanejvýš důležitý (!!) závěr: absolutní bezpečnost neexistuje!

4. Slabé místo, ohrožení a riziko

- Slabé místo/zranitelnost (angl. Weakness/vulnerability) slabinou systému nebo bodem, v němž může být systém zranitelný (prostřednictvím zneužití).
- hrozby (angl. threat) plynou z možných útoků, které využívají jedno či několik slabých míst systému za účelem ohrožení jednoho či několika cílů ochrany.
- riziko R (angl. risk) hrozby je pravděpodobnost E výskytu škodné události a výše potenciální škody S, která z toho může vzniknout: $R = E \cdot S$

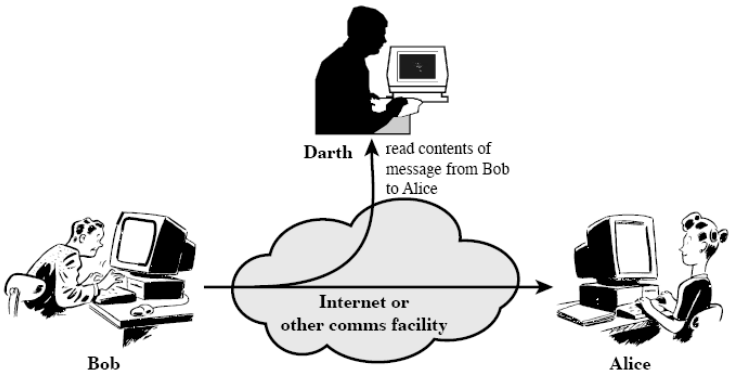
4.1. Souvislosti



5. Kategorie, úrovně & příčiny útoku

Pasivní útoky – Odposlech (Eavesdropping)

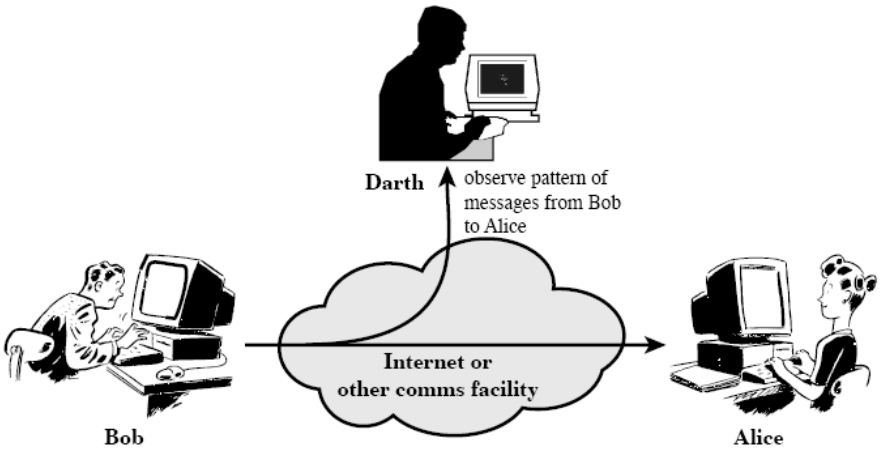
- Útočník odposlouchává komunikační kanál, aktivně však nezasahuje do komunikace.



(a) Release of message contents

Pasivní útoky – Analýza toku dat (Traffic Analysis)

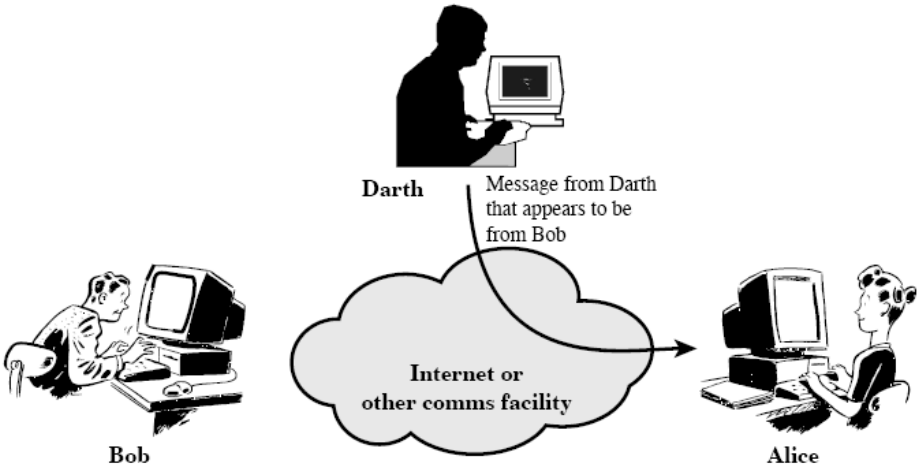
- U zakódovaného datového kanálu může pasivní útočník provádět analýzu toku dat (Traffic Analysis) (kdo komunikuje kdy s kým?)



(b) Traffic analysis

Aktivní útoky – zamaskování

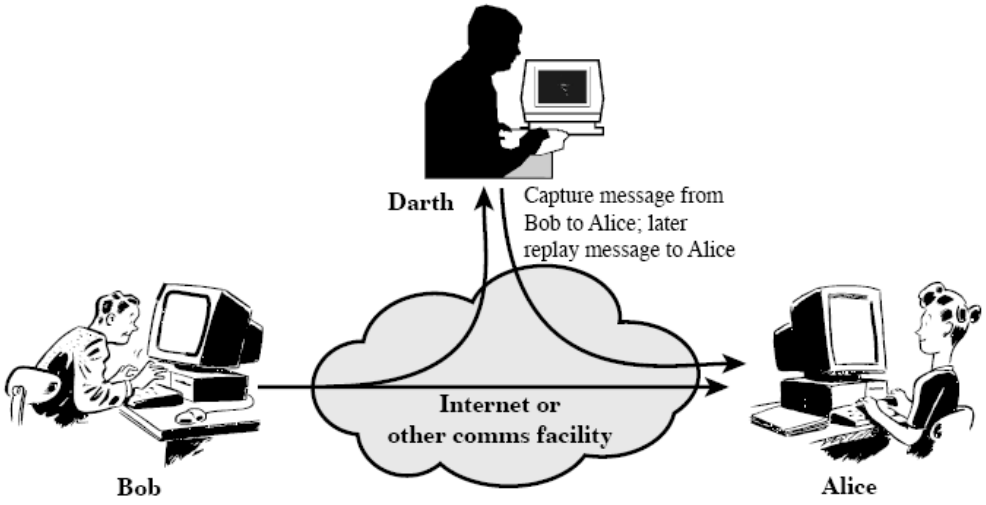
- Zamaskování: útočník se vydává za někoho jiného.



(a) Masquerade

Aktivní útoky – Insertion & Replay

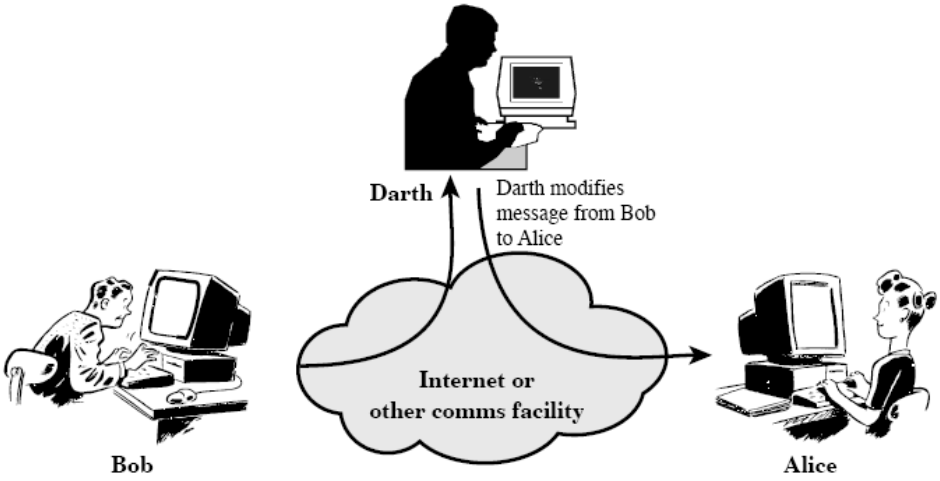
- Insertion: útočník přidává zprávy (části zpráv) k nějaké komunikaci.
- Replay: útočník odesílá zaznamenaná data později ještě jednou.



(b) Replay

Aktivní útoky – Modification

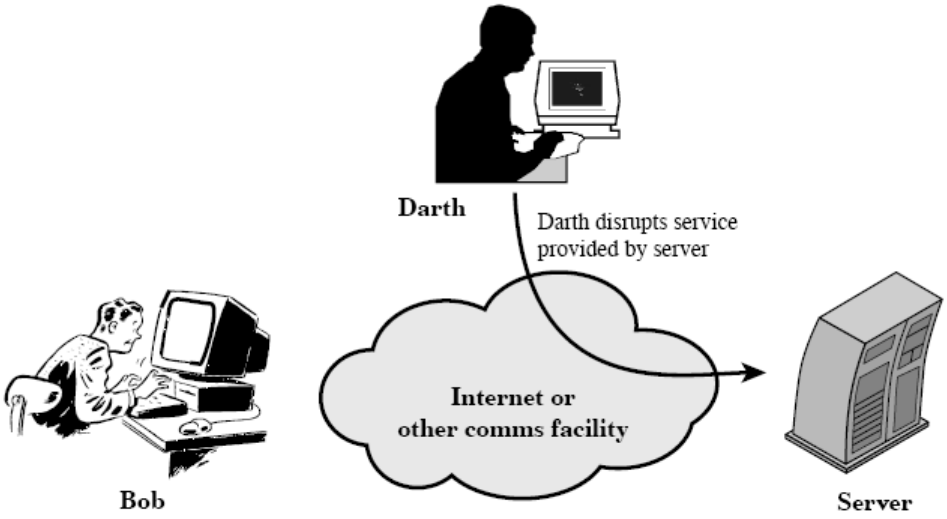
- Útočník pozmění komunikaci tím, že pozdrží, změní či smaže zprávy.



(c) Modification of messages

Aktivní útoky – Denial of Service

- Útočník narušuje dostupnost komunikačních zařízení



(d) Denial of service

5.1. Úrovně útoku aktuálně (výtah)

- síť
  - botnety
  - (Distributed) Denial of Service (odepření služby), Reflection, Amplification
  - Spam (Unsolicited Commercial/Bulk E-Mail, Social Media Spam)
  - útoky Man-in-the-Middle (např. k spolučtení/modifikaci komunikace)

- aplikace (především webové aplikace, viz OWASP Top 10), např.
  - Injection (z. B. SQL Injection, Command Injection)
  - Cross-Site Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)
  - Defacements
  - Buffer Overflows

- uživatel
  - Social Engineering
  - (Spear) Phishing
  - Scareware, Ransomware
  - Spam

5.2. Příčiny

- chybějící/nedostatečné ověření identity
  - např. slabé postupy autentifikace, unilaterální autentifikace, …

- chybějící/ nedostatečná validace vstupu
  - např. údaje uživatele ve webových aplikacích nejsou ze strany serveru ověřovány

- psychologické nedostatky a překážky
  - např. Social Engineering

- faktory nákladů a času ve vývoji produktu
  - např. krátké cykly distribuce, chybějící testy, bezpečnost není zahrnuta do vývojových procesů

- organizační nedostatky
  - např. chybějící bezpečnostní management, chybějící povědomí uživatele, …

6. Typy útočníků & jejich motivace

- amatéři (Script Kiddies)
  - zodpovědní za většinu (automatizovaných) útoků
  - jen málo z nich má hlubší znalosti → aplikace hotových nástrojů útoku
  - motivy: prestiž, osobní msta, nuda

- Crackers vs. Hackers (Blackhats vs. Whitehats)
  - Crackeři jsou zlí hackeři (pojmosloví nejasné)
  - hluboké technické znalosti (studenti, informatici)
  - motivy: prestiž, intelektuální výzva

- zločinci (Počítačová kriminalita)
  - klasičtí zločinci, kteří mění povolání jen z důvodu zisku
  - Spamming, online vydírání, Bulletproof-Hosting/-Services, botnety (pronájem), ...
  - dobře organizované sítě s napojením na organizovaný zločin (např. Russian Business Network, Silk Road, ...)
  - motivy: zisky

- teroristé (Cyber-Terrorismus)
  - IT jako cíl útoku → poškodit/zničit infrastrukturu cíle
  - účely propagandy
  - komunikace a organizace

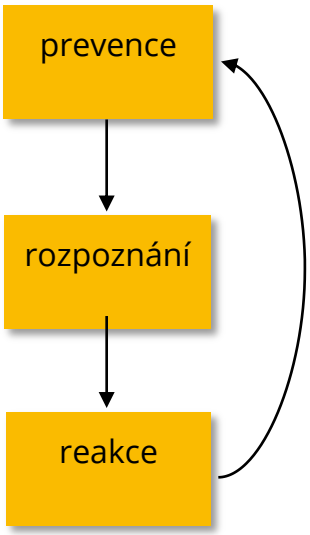
- země a její vojenské/zpravodajské aparáty (Cyber-War(fare), Cyber-Espionage)
  - odstavení IT infrastruktury jako cíl vojenských útoků
  - hospodářská, politická a vojenská špionáž (→ Čína, USA)
  - internet jako zbraň
  - příklady: Stuxnet, Flame, Regin, skandál NSA se sledováním (špehováním)

7. Klasifikace bezpečnostních opatření

- opatření k zamezení (a priori)
  - např. autentifikace, kontrola přístupu, použití zakódování, Firewalls, stability systémů,
    koncepce bezpečnosti, politika bezpečnosti, vytvářet povědomí, ...

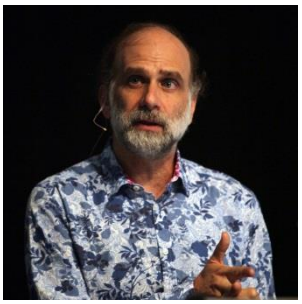
- opatření k rozpoznání
  - dynamicky době trvání
  - např. Firewalls, Intrusion Detection Systeme, Log-Analyse, ...

- opatření k omezení škody (a posteriori)
  - např. zpřísnění kontrol, zlepšení bezpečnostních opatření, kryty internetového připojení, ...



8. Kontinuální komplexnost v informační bezpečnosti

Bezpečnost není produkt, je to pro-



ces!

Bruce Schneier

Photograph by Rama, Cc-by-sa-2.0-fr, from Wikimedia Commons

- Komplexnost je kritický faktor úspěchu pro IT/informační bezpečnost
  - např. Firewall k filtrování toku dat, avšak uživatelé mohou k síti připojit WiFi Access Points
  - např. přístupy k systémům jsou chráněny hesly, která jsou zveřejněna na informační tabuli
  - např. uživatelé si poznamenají svá hesla k systému Windows na samolepicích barevných papírcích na svých monitorech

- IT/informační bezpečnost vyžaduje souhru technických a organizačních opatření!

- za IT/informační bezpečnost v podniku musí zodpovídat management!

- IT/informační bezpečnosti je třeba se věnovat kontinuálně a zdokonalovat ji (příklady?)!

8.1. Bezpečnostní standardy

- bezpečnostní management/bezpečnostní proces, např.
  - ISO 27000er rodina
  - BSI standardy základní ochrany 100-1, 100-2, 100-3 und 100-4

- bezpečnost systému (certifikace produktů), např.
  - TCSEC (Trusted Computer System Evaluation Criteria)
  - ITSEC (Information Technology Security Evaluation Criteria)
  - Common Criteria for Information Technology Security Evaluations (ISO 15408)

- katalog opatření (technická/organizační), např.
    - BSI katalogy základní ochrany

- další relevantní standardy ([IT]-Governance, Compliance), např.
    - COBIT (Control Objectives for Information and Related Technology)
    - ITIL (IT Infrastructure Library)

8.2. Právní normy / zákony

- Rakouská strategie kybernetické bezpečnosti (ÖSCS)
    - v budoucnu: Rakouský zákon o kybernetické bezpečnosti

- směrnice EU o opatřeních k zajištění vysoké společné úrovně bezpečnosti sítí a informačních systémů (směrnice NIS)
    - implementace do Rakouského zákona o kybernetické bezpečnosti

- Obecné nařízení o ochraně osobních údajů, EU

9. Literatura

[Eckert2008] Eckert, C., Vorlesungsskript zur VO IT-Sicherheit, TU Darmstadt, SS 2008

[Stallings2006] Stallings, W., Cryptography and Network Security, 4th edition, Prentice Hall, 2006

[BSI1992] Bundesamt für Sicherheit in der Informationstechnik (BSI), IT-Sicherheitshandbuch – Handbuch für die sichere Anwendung der Informationstechnik, Version 1.0, März 1992,

[BSI2005] Bundesamt für Sicherheit in der Informationstechnik (BSI), IT-Grundschutzhandbuch, Stand 2005

SÍTĚ

1. Základní pojmy

V této části se pokusíme na začátku nastínit možnou definici základních pojmů, se kterými se mohou studenti v oblasti počítačových sítí setkat. Je třeba říci, že zde uvedené popisky není možné brát jako dokonalé definice – jde o velice komplexní oblast, která téměř vždy umožňuje existenci určitých výjimek. Je ale třeba, aby měli studenti alespoň základní představu o těchto pojmech a uměli jich správně užívat.

Klient je označení počítače, který je připojen do sítě. Nemá obvykle žádné privilegované postavení a jeho možnosti jsou podobné jako jakéhokoli jiného klienta. Často se také můžeme setkat s tímto pojmem jako s označením softwaru, který komunikuje se serverem, a zprostředkuje tak uživateli nějakou službu (FTP klient, e-mailový klient, …).

Server je počítač, který má v síti určité výsadní postavení. Jeho úkolem je zajistit klientům určité služby či funkce – může jít o vzájemnou komunikaci klientů, převod doménového jména na IP adresu (DNS), připojení na internet, tisk, sdílení souborů atp. Server může vystupovat ve vztahu k jinému serveru v roli klienta. Obvykle jde o počítače, které jsou specifické také svojí konstrukcí (speciální procesor, disková pole, …).

Směrovač je zařízení, které zajišťuje směrování paketů. Jde o aktivní prvek sítě, který se snaží mít znalost o svém okolí a příchozí pakety nasměrovat správným směrem. V analogii s doručováním paketu jako dopisu by plnil roli pošty. Jeho využití je ale širší – umožňuje garantovat kvalitu služeb, hlídá TTL (jde o parametr omezující životnost paketů, aby ztracený paket nebloudil sítí věčně), musí aktivně zkoumat své okolí a zjišťovat změny v síti. Současně provádí výpočet nejlepší cesty k určitým dalším uzlům.

Paket představuje pevně definovaný balíček dat, který je posílán sítí. Data nejsou v počítačových sítích obvykle (téměř nikdy) posílána jako souvislý tok informací, ale v určitých balíčcích, tedy paketech. To umožňuje vyšší bezpečnost, lepší směrování i efektivnější využití kapacity sítě. Paket má pevně danou strukturu, ale obecně je možné říci, že vždy obsahuje adresu odesílatele a příjemce, informaci o používaném protokolu, vlastní data a případně nějaké další informace.

Síťová karta je hardwarové zařízení (dnes nejčasněji integrované na základní desce počítače), které zajišťuje zapojení počítače do sítě. Informace jsou v ní převáděny na pakety, a naopak pakety skládány do dat. Karta umí specifický protokol přenosu dat (typicky Ethernet) a má port pro zapojení kabelu (kroucená dvojlinka, optické vlákno, koaxiální kabel). Mimo to má specifickou MAC adresu, která je jedinečná v rámci celého světa a slouží pro identifikaci počítače v síti a případně vygenerování lokální adresy.

Port je číslo v rozsahu od 0 do 65535, které identifikuje aplikaci, jenž se právě účastní komunikace. Jednotlivé aplikace se identifikují v rámci protokolu TCP či UDP. Díky portům je možné provádět jednoduchou obsluhu aplikací, zajišťovat (do určité míry)

kvalitu služeb atp. Existují obvyklá čísla portů, např. SMTP má 25, POP3 110, HTTP 80 nebo 21 patří k FTP. Uživatelé či aplikace se ale mohou shodnout na jiném portu, což se však obvykle příliš nedělá.

Médium je fyzické prostředí, prostřednictvím kterého probíhá síťová komunikace. Můžeme mít média voděná – pak jde o klasickou kabeláž (optický kabel, matalický kabel, kroucená dvojlinka) nebo vlněná (vzduch, vakuum), která využívají technologie, jako je WiFi či GSM.

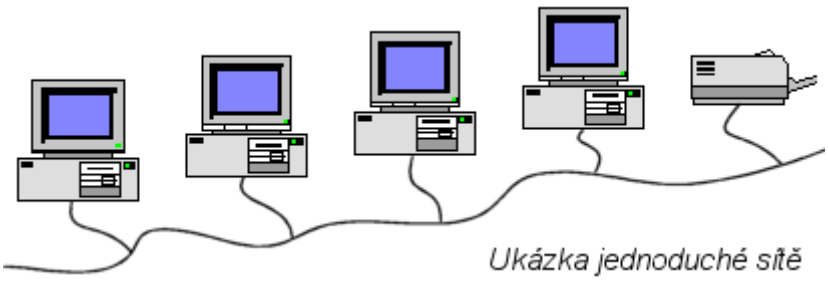
Protokol je formální jazyk definující způsob komunikace (IP, TCP, IMAP4). Přesně popisuje, jak vypadá paket, jakou rychlostí se má zasílat, definuje bezpečnostní prvky, opravy chyb atp. Téměř každá vrstva ISO-OSI modelu má vlastní skupinu protokolů. Většinou jsou vázány pouze na jednu vrstvu.

Switch (česky přepínač) je aktivní prvek sítě, které propojuje její jednotlivé části. Na rozdíl od mostu (bridge) umožňuje spojovat více než dva síťové okruhy. Menší části sítě mají výhodu v tom, že disponují lepším přístupem k médiu, čímž je zajištěna větší přenosová rychlost a využití sítě.

Opakovač je jednoduché zařízení, které má za úkol zajistit přenos signálu na větší vzdálenosti. Je umístěn v určité vzdálenosti od vysílače a zesiluje příchozí signál. Pracuje tedy přímo na fyzické vrstvě.

1.1. K čemu slouží počítačová síť?

Společnosti si instalují počítačové sítě především proto, aby mohli sdílet zdroje a aby umožnili přímou komunikaci. Zdroje zahrnují data, aplikace a periferní zařízení. Periferním zařízením je například externí disketová mechanika, tiskárna nebo modem. Přímá komunikace zahrnuje posílání zpráv, odpovídání na zprávy nebo e-mail.



Ukázka jednoduché sítě

Počítačová síť je souhrnné označení technických prostředků, pomocí nichž se realizuje propojení a výměna dat mezi počítači. Umožňuje uživatelům komunikaci podle zadaných pravidel. Nejčastějším důvodem připojení k síti je sdílení informací a technických zařízení.

Obrázek 1: zapojení sítě do hvězdy

Podle rozlehlosti se rozlišují čtyři počítačové sítě:

- **PAN** (původem v anglickém ***Personal Area Network***), síť pokrývající velice malé území (menší než území LAN), síť tvořená komunikujícími zařízeními, jako třeba mobil a notebook,
- **LAN** (původem v anglickém ***Local Area Network***), **lokální síť** – též zvána **místní síť** nebo síť pokrývající malé území, území větší než území PAN, ale menší než území MAN,
- **MAN** (původem v anglickém ***Metropolitan Area Network***), **metropolitní síť** – též zvána, síť na území větším než území LAN a menším než území WAN,
- **WAN** (původem v anglickém ***Wide Area Network***) –, síť pokrývající rozlehlé území, území větší než území MAN;
- **VLAN** je takzvaná virtuální LAN (síť). V rámci funkční sítě LAN je tedy vytvořená další, virtuální síť, která funguje i v rámci jen jednoho hardwarového zařízení (kabelu). VLAN funguje na principu značkování dat, které se po síti posílají. Data se označují podle virtuální sítě, do které patří.

2. Topologie sítě

Topologie sítě je způsob jejího fyzického zapojení. Pokud bychom chtěli být zcela korektní, bylo by nutné rozlišit mezi logickým a fyzickým propojením počítačů.

Kruhová topologie je propojení počítačů do kruhu. K jednomu počítači je připojena vždy dvojice dalších. Výhodou tohoto modelu je, že pokud vypadne jeden prvek, síť může (pokud je duplexní) fungovat dál. Stejně tak je respektován fakt, že nejbližší počítače

spolu (v LAN) obvykle komunikují nejčastěji.

Mesh je zřejmě nejméně používanou topologií. Nejsou v ní žádná pravidla, je budována ad hoc, tak jak jsou postupně přidávány počítače. Na druhou stranu je zajímavá z hlediska popisu, neboť je zcela obecná. Z logického uspořádání je možné říci, že Mesh je internet nahlížený jako celek.

Hvězda představuje jeden z nejčastějších modelů zapojení – jednotlivé stanice jsou připojené k jednomu centrálnímu bodu (například serveru). Mohou spolu komunikovat výhradně přes něj. Výhodou je snadná správa a směrování, slabým místem naopak server.

Strom představuje další topologii, která patří mezi užívané. Existuje zde určitá struktura počítačů, které jsou zapojeny do struktury matematického stromu. Jednotlivé počítače tak mohou (nejsou-li koncové ani kořenové) plnit roli serveru a klienta současně.

Sběrnice představuje model, kdy k jednomu médiu jsou postupně připojeny stanice. Jde tedy o koncept sdíleného média.

Existují samozřejmě také další topologie – lineární, spojení každého s každým, nebo extrémní případ dvoubodového spojení. Většinou je to něco mezi tím.

2.1. Počítačové sítě a jejich služby

Počítačová síť je souhrnné označení technických prostředků, pomocí nichž se realizuje propojení a výměna dat mezi počítači. Umožňuje uživatelům komunikaci podle zadaných pravidel. Nejčastějším důvodem připojení k síti je sdílení informací a technických zařízení.

Obr.: Počítačová síť

2.2. Síťové topologie

Topologie je způsob uspořádání počítačů do sítě. Podává nám informace o tom, jak jsou počítače propojené a jak spolu komunikují. Možné způsoby zapojení jsou na obrázcích níže.

Sběrnicová topologie Hvězdicová topologie Kruhová topologie Neomezená topologie

Obr.: Topologie - možná uspořádání počítačů v síti

U sběrnicové topologie spojení zajišťuje přenosové médium nazývané sběrnice, ke kterému jsou připojeny všechny koncové počítače, resp. uzly sítě.

Hvězdicová topologie označuje propojení počítačů do tvaru hvězdy. Je to nejpoužívanější způsob propojování počítačů do počítačové sítě. Základním kamenem je počítač uprostřed, kterému se říká centrální počítač. S ním jsou spojeny všechny ostatní počítače okolo něj. Lze si představit, že jeden nebo více koncových počítačů jsou nahrazeny centrálním počítačem. Celá síť se tak neomezeně zvětšuje.

Kruhová topologie označuje zapojení, kde je jeden uzel sítě připojen k dalším dvěma uzlům tak, že vytvoří kruh. Tato topologie není příliš rychlá, protože při velkém počtu uzlů bude trvat, než přenášená data dorazí do cílového počítače. Pokud navíc jeden z uzlů nefunguje, není možné data přenést a celá síť přestane být funkční. Toto zapojení se dnes prakticky nevyužívá, protože hvězdicová topologie je mnohem spolehlivější. Na druhou stranu náklady na vybudování této sítě jsou menší.

3. Co je to internet?

Internet je celosvětová počítačová síť podobná klasické počítačové síti. Počítače jsou mezi sebou vzájemně propojeny a díky tomu spolu mohou komunikovat a sdílet informace. Internet je propojením již stávajících sítí, které mají určitou strukturu a rozdělení.

Internet můžeme také definovat jako soustavu počítačů, které obsahují informace a sítě, které nám dovolují k těmto informacím přistupovat.

Počítače v internetu pracují jako klienti a servery. Servery poskytují internetové služby, klienti pak tyto služby využívají. Službami Internetu je zasílání dat ke klientovi na jeho žádost.

3.1. Vznik internetu

V době studené války potřebovalo USA fungující systém řízení a velení, který by byl schopný spojit nejdůležitější akademické, vládní a strategické počítače (státy,

vojenské základny,...). Požadavkem bylo vytvořit odolnou síť, která bude funkční i po výpadku několika uzlů a stále bude zajištěna, byť alespoň částečná komunikace.

Počátkem 60. Let 20. Století byla firma RAND Corporation požádána o vyřešení problému, který spočíval ve fungování počítačů i po jaderné válce. Úkolem bylo nalézt, vymyslet fungující systém i přesto, že by některé jeho části mohly být zničeny.

V roce 1964 přišla stejná firma s řešením, které bylo založeno na dvou principech:

- Síť nebude mít žádnou centrální složku
- Síť bude fungovat, i když jsou některé části mimo provoz

Jako tvůrce internetu můžeme tedy jednoznačně označit americkou armádu, potažmo Pentagon. V roce 2010 se eviduje na 2 miliardy aktivních uživatelů internetu.

3.2. Vývoj internetu

Nultá fáze

Označuje vznik internetu díky potřebě americké armády na komunikaci mezi vládními složkami, kdyby došlo k jaderné válce v průběhu studené války. Ministerstvo obrany USA, resp. agentura ARPA (Advanced Research Projects Agency) začala vyvíjet síť ARPANET a řídilo vývoj i financování tohoto projektu.

První fáze

V první fázi měla síť na dálku umožnit přístup k tehdejším nejvýkonnějším počítačům, především univerzit v USA. Koncem roku 1969 byly první uzly sítě ARPANET umístěny právě na univerzitách. Snahou připojení k síti nebylo připojit pouze jeden počítač, snahou bylo připojit celou síť. Rozvoj proběhl ve všech možných organizacích, začínaly se používat lokální počítačové sítě. Počátek 80. Let zaznamenal rozvoj ARPANETu a pokračoval i v jiných sítích. Vznikly sítě jako třeba Usenet a BITnet. Vše ve výsledku směřovalo k propojení s ARPANETem. V roce 1987 bylo možné registrovat více než 10 000 připojených uzlů, o 2 roky později to již bylo více než desetinásobek. 80. a 90. Léta odstartovala služby, které již dnes běžně známe a používáme, jednalo se především o email (1971), telnet (1972), TCP (1974) a jeho specifikace na TCP/IP (1978), DNS (1983) a jeho start (1984).
V roce 1980 Pentagon rozhodl, že preferovanými protokoly pro rezort obrany budou právě protokoly TCP/IP a o 2 roky později jsou všechny počítače připojené k síti ARPANET nuceni přejít na protokoly TCP/IP. ARPANET se tak stal zárodečnou sítí a vznikal konglomerát vzájemně existujících a nově vznikajících sítí, který byl označován jako Internet.

Druhá fáze

Druhá etapa označuje rozvoj internetu v období 1983 – 1992. Období je charakteristické prudkým růstem Internetu a především expanzí mimo americký kontinent. Na ARPANET se napojují další sítě, jako třeba NFSNET, EUNET, EARN, JUNET, apod. Listopad 1983 byl právě ten den, kdy byl zaveden doménový systém DNS, který umožnil číselným adresám přidělovat doménová jména. Roku 1989 je vynalezen WWW (World Wide Web), který se následně stává nedílnou součástí internetu. Roku 1990 dochází k odstavení ARPANETu a následně k jeho zrušení. Páteřní sítí Internetu se stává NFSNET. Roku 1991 dochází k připojení České Republiky do internetu. Od roku 1993 dochází k uvolnění internetu i mezi běžné uživatele

3.3. K čemu internet slouží

Díky internetu mohou jeho uživatelé využívat nespočetné množství služeb. Služby jsou zajišťovány počítačovými programy a programy, které komunikují navzájem pomocí protokolů. Uvedeme si základní služby internetu.

- **WWW** – systém webových stránek zobrazovaných pomocí internetového prohlížeče.
  Používá protokol http/HTTPS
- **E-mail** – elektronická pošta
  Odesílání zpráv přes protokol SMTP
  Přijímání zpráv skrze protokol POP3, IMAP
- **IM – Instant messaging** – online, živá komunikace ICQ, Jabber, Skype
- **FTP** – přenos souborů
- **DNS** – systém jmen počítačů pro jejich snadnější zapamatování

Dále můžeme jmenovat nespočet dalších protokolů, např. sdílení souborů (NFS), protokoly sloužící k připojení ke vzdálenému počítači (telnet, SSH, VNC) a jiné.

3.4. Ethernet

Ethernet je technologie pro počítačové sítě. Ethernet představuje souhrn technologií, které se zabývají přenosem dat především v LAN sítích. Většina jeho verzí podléhá standardizaci institutu IEEE. Ethernet je realizací první (fyzické) a druhé (spojové) vrstvy referenčního modelu ISO/OSI. V rámci Ethernetu se jako přenosové médium nejčastěji používá kroucená dvojlinka či optický kabel, v prvotních verzích se pracovalo i s kabely koaxiálními. Nezbytnou součástí je konektor RJ-45. Existuje několik typů Ethernetu, v závislosti na druhu kabeláže a výše přenosové rychlosti. Ta se v různých verzích pohybovala od 10 Mbit/s až po 100 Gbit/s.

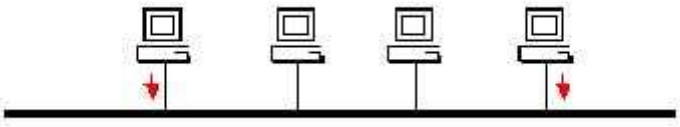
V současné době je nejpoužívanější síťovou technologií Ethernet. Tato technologie je, nezávisle na tom zda jde klasický 10 Megabitový Ethernet nebo jeho rychlejší mutace

(Fast a Gigabit Ethernet), založena na velice jednoduchém principu, nazývaném CSMA/CD.

Na strukturovaném kabelovém systému lze používat rozličné síťové technologie založené na rozdílných přenosových metodách - např. Ethernet, Token Ring, CDDI, ATM, ...

V současné době je nejpoužívanější síťovou technologií Ethernet. Tato technologie je, nezávisle na tom, zda jde klasický 10 Megabitový Ethernet nebo jeho rychlejší mutace (Fast a Gigabit Ethernet), založena na velice jednoduchém principu, nazývaném CSMA/CD.

CSMA (Carrier Sense Multiple Access) - stanice připravená vysílat data si "poslechne" zda přenosové médium (kabel) nepoužívá jiná stanice. V případě, že ano, stanice zkouší přístup později až do té doby dokud není médium volné. V okamžiku, kdy se médium uvolní, začne stanice vysílat svá data.



CD (Collision Detection) - stanice během vysílání sleduje, zda je na médiu signál odpovídající vysílaným úrovním (tedy aby se např. v okamžiku, kdy vysílá signál 0, nevyskytl signál 1). Případ, kdy dojde k interakci signálů více stanic, se nazývá kolize. V případě detekce kolize stanice generuje signál JAM a obě (všechny) stanice, které v daném okamžiku vysílaly, generují náhodnou hodnotu času, po níž se pokusí vysílání zopakovat.

Díky této jednoduchosti bylo dosaženo nízké ceny síťových adaptérů a aktivních prvků a tím i značného rozšíření Ethernetu. Jednoduchost řešení ovšem přináší i jednu významnou nevýhodu – s narůstajícím počtem uzlů narůstá počet kolizí a tím klesá teoretická propustnost sítě. Soubor uzlů, jejichž vzájemná činnost může vygenerovat kolizi, se nazývá **kolizní doména**. Logicky lze odvodit, že kolizní doména by měla být co nejmenší. Používané aktivní prvky mají ke kolizní doméně rozdílný vztah. Některé kolizní domény rozšiřují, některé kolizní domény oddělují. Jejich volbou lze proto propustnost sítě ovlivnit.

Vedle pojmu kolizní doména existuje pojem **broadcastová doména**. Na počítačové síti se vyskytují principielně dva typy paketů – tzv. unicasty a nonunicasty. Unicasty jsou pakety, které mají konkrétního adresáta vyjádřeného regulérní síťovou adresou. Nonunicasty používají skupinovou adresu a jsou určené buď všem uživatelům sítě (broadcasty) nebo vybrané skupině uživatelů (multicasty). Problém je v tom, že nonunicastu se musí počítač věnovat i když není určen pro něj. S nárůstem počtu uzlů v broadcastové doméně narůstá i množství nonunicastů. Z tohoto důvodu je nutné udržet velikost broadcastové domény v rozumné velikosti. Používané aktivní prvky mají k broadcastové doméně rozdílný vztah a proto lze jejich volbou propustnost sítě ovlivnit.

Formát paketu

Již bylo řečeno, že všechny rychlostní modifikace Ethernetu používají stejnou komunikační metodu CSMA/CD. Používají však i stejný formát a velikost paketu. Ethernetový paket je definován na 1. a 2. vrstvě OSI.

4. Standardizace počítačových sítí, zásobník TCP/IP

4.1. Standardizace počítačových sítí

Původní strategie významných výrobců spočívala ve vytváření a udržování takových specifikací, které zajišťovaly uživatelům kompatibilitu výhradně vlastních výrobků firmy. Tyto počítačové sítě označované jako homogenní nebo *uzavřené systémy* měly nevýhodu v tom, že se jim museli přizpůsobovat jak výrobci technického vybavení, tak i výrobci programového vybavení (vyjmenujme alespoň některé ze známých vlastnických protokolů - *DECnet* od firmy DEC, *SNA* - Systems Network Architekture od firmy IBM, *SPX/IPX* - Sequenced/Internet Packet Exchange od firmy Novell atd..). Na rozdíl od těchto uzavřených systémů *otevřené* neboli *heterogenní systémy* jsou počítačové systémy, které jsou navrženy v souladu s uznávanými mezinárodními standardy a lze je získat od více nezávislých výrobců. Užití otevřených systémů umožňuje, aby programy pracující v lokálních nebo vzdálených systémech sítě mohly navzájem spolupracovat a aby komunikace uživatele s těmito aplikacemi byla jednotná.

Pojem propojení otevřených systémů (OSI - *Open Systems Interconnection*) označuje technické a programové vybavení počítačové sítě. V oblasti počítačových sítí jsou mezinárodní organizace CCITT (Comité Consultatif International de Télégraphique et Téléphonique - je jednou ze tří stálých komisí ITU - International Telecomunications Union) a ISO (International Organization for Standardization) nejdůležitější a zabývají se tvorbou standardů. Členy CCITT jsou především vládní instituce nebo organizace, které zodpovídají v jednotlivých členských zemích za telekomunikace. CCITT vydává doporučení, které se po schválení ITU stávají mezinárodními standardy. Na rozdíl od této organizace je ISO dobrovolnou nevládní organizací, jejími členy jsou normalizační instituce. Další důležité a známé jsou např. IEEE (Institute of Electrical and Electronics Engineers), což je prestižní americká společnost elektrotechnických a elektronických inženýrů. Tato společnost vydává odborné časopisy, pořádá konference a má vlastní orgán, který se zabývá tvorbou standardů. Pro oblast lokálních počítačových sítí je známá skupina standardů IEEE 802. V roce 1979 přijala organizace ISO standard pod označením Reference Model of Open Systems Interconnection (RM OSI).

Síťová komunikace

Činnost sítě znamená posílat data z jednoho počítače do druhého. Tento složitý problém se dá rozdělit do těchto úkolů:

- rozpoznat data,
- rozdělit data do zpracovatelných dávek,
- připojit ke každé dávce dat informace (zdroj a cíl),
- připojit informace o časování a kontrole chyb,

- převést data do sítě a poslat je do určeného místa.

4.2. Základní pojmy RM OSI

Vrstva (*Layer*) - je dána přesným vymezením funkcí. Ke každé vrstvě, kromě nejvyšší a nejnižší, přísluší rozhraní s bezprostředně vyšší a nižší vrstvou. Nejvyšší vrstvě náleží rozhraní s aplikačním procesem. Nejnižší vrstva pak sdílí rozhraní s fyzickým médiem.

Entita (*Entity*) - je objekt, který v určité vrstvě něco aktivního vykonává. Obvykle se jedná o programový celek. U nižších úrovní jde o hardwarový celek (I/O port). Bezprostřední poskytovatel a uživatel služby není vrstva, ale právě určitá entita vrstvy.

Protokol (*Protocol*) - entity ležící ve stejných vrstvách různých otevřených systémů spolu **komunikují** pomocí předem dohodnutých pravidel, kterým říkáme protokoly.

Služba (Service) - entity určité vrstvy, když vykonávají své funkce, poskytují služby bezprostředně vyšší vrstvě, pro realizaci těchto funkcí využívají služeb vrstvy bezprostředně nižší. Jednotlivé služby jsou nabízeny v tzv. bodech poskytování služby (Service Access Point, SAP), které jsou identifikovány prostřednictvím svých adres. Služby lze rozdělit na:

- **Spojové služby** (*Connection - oriented Services*) - kde dvě entity na stejných úrovních, když chtějí komunikovat, musí nejdříve navázat spojení. Po spojení odesílající posílá, co si přeje odeslat a **příjemce** si na druhé straně zprávu přebere. Jde o obdobu navázání jako u telefonního spojení.
- **Nespojované služby** (*Connectionless Services*) - nepočítají se zřízením stálého spojení mezi **odesílatelem** a příjemcem. **Místo** toho se považují jednotlivé části přenášených dat **za** samostatné celky opatřené adresou svého konečného příjemce a doručují se nezávisle na ostatních zprávách. Jednotlivé zprávy mohou být přenášeny různými cestami.
- **Spolehlivá služba** (*Reliable Service*) - je taková, která nikdy neztrácí žádná data. Obvykle je to **služba** realizována prostřednictvím vhodného mechanismu potvrzování zpráv.
- **Nespolehlivá služba** (**Unreliable** *Service*) - kde není potvrzování zpráv, ale přesto se jedná o službu s velmi vysokou spolehlivostí.

4.3. Referenční model ISO/OSI

Tento model byl vytvořen mezinárodní organizací ISO jako jednotný standard pro vzájemné propojování systémů různých typů a koncepcí, pocházejících od různých výrobců. Vzniklý model má sedm vrstev. Sedm vrstev tvoří hierarchii začínající aplikacemi na vrcholu a končící fyzickými spojeními vespod. Referenční model OSI zahrnuje dva modely komunikace:

- horizontální, model na protokolové bázi, pomocí něhož komunikují programy a procesy různých počítačů,
- vertikální, model na bázi služeb, pomocí něhož komunikují vrstvy na jediném počítači.

Komunikace vyžaduje následující prvky. Alespoň dvě strany mající zájem komunikovat a musí existovat společný jazyk (protokol) pomocí kterého budou strany komunikovat. Vertikální vrstvy komunikují přes rozhraní API (Application Program Interface).

4.3.1. APLIKAČNÍ VRSTVA (APPLICATION LAYER)

Tato vrstva specifikuje prostředí, ve kterém síťové aplikace komunikují se síťovými službami. Koncový uživatel využívá sítě pro spouštění aplikací, přenos souborů, poštu, pro vzdálené přihlášení (remote login) apod.. Protokoly aplikační vrstvy obsahují především aplikační programy. Nalezneme zde i síťové nadstavby, které umožňují připojení stanice k síti. Mezi programy a protokoly poskytující služby aplikační vrstvy patří následující:

- NICE (Network Information and Control Exchange), který poskytuje monitorování a správu sítě.
- FTAM (File Transfer Access and Management), který poskytuje vzdálenou správu souborů.
- X.400, který specifikuje protokoly a funkce pro předávání zpráv a elektronickou poštu.
- CMIP, který nabízí správu sítě postavené na rámci formulovaném OSI.
- Telnet, který poskytuje terminálovou emulaci a vzdálené připojení.
- rlogin, který poskytuje vzdálené připojení pro prostředí UNIX.

4.3.2. PREZENTAČNÍ VRSTVA (PRESENTATION LAYER)

Tato vrstva řídí formátování datových přenosů. Data, která prostřednictvím sítě se přenášejí, mohou být texty, čísla nebo obecnější datové struktury. Jsou zde obsažena specifika pro kódování a dekódování znakových sad, v rámci této vrstvy bývá realizována i komprese dat nebo jejich šifrování atd.. Prezentační vrstva poskytuje služby pro aplikační vrstvu ležící nad ní a využívá relační vrstvu pod ní.

4.3.3. RELAČNÍ VRSTVA (SESSION LAYER)

Tato vrstva poskytuje procesy, které řídí přenos dat, zpracovává chyby vysílání a přenosu, vede záznamy o vyslaných přenosech. Jinak tato vrstva navazuje, udržuje a ruší relace mezi koncovými účastníky (mezi uživatelem a cílovým počítačem). V rámci navazování relace se tato vrstva vyžádá na transportní vrstvě vytvoření spojení, jehož prostřednictvím pak probíhá komunikace mezi oběma účastníky relace. Rovněž určuje, kdo má kdy vysílat a ruší existující spojení. Protokoly relační vrstvy:

- ADSP (AppleTalk Data Stream Protocol), který umožňuje, aby dva uzly vytvořily spolehlivé spojeni pro přenos dat.
- NetBEUI, který je implementací a rozšířením NetBIOSu.
- NetBIOS, který využívá vrstev 5. 6 a 7, nabízí i služby monitorování relací.
- PAP (Printer Access Protocol), který poskytuje přístuů k tiskárně PostScript v síti AppleTalk.

4.3.4. TRANSPORTNÍ VRSTVA (TRANSPORT LAYER)

Tato vrstva poskytuje funkce pro vytvoření příslušných spojení, rozkládá data na menší části, tzv. pakety, do kterých rozděluje přenášená data a při příjmu je zase z paketů vyjímá a skládá do původního tvaru. Dokáže tak zajistit přenos libovolně velkých zpráv, přestože jednotlivé pakety mají omezenou velikost. Transportní vrstva je z mnoha důvodů nepostradatelná, protože sídlí mezi horními vrstvami a nižšími vrstvami, které jsou síťově orientované.

4.3.5. SÍŤOVÁ VRSTVA (NETWORK LAYER)

Tato vrstva popisuje procesy, které směřují data mezi síťovými adresami a kontrolují, zda zprávy byly poslány kompletní a včas. Jinak zajišťuje potřebné směrování (routing) přenášených rámců označovaných nyní jako pakety (packets). Musí si uvědomovat

konkrétní topologii sítě, tzn. způsob vzájemného přímého propojení jednotlivých uzlů.

4.3.6. LINKOVÁ VRSTVA (DATA LINK LAYER)

Tato vrstva, která se často nazývá jako spojová vrstva, se zabývá procesy, které detekují a opravují chyby na úrovni dat během datového přenosu mezi fyzickou vrstvou a vrstvami nad fyzickou vrstvou. Linková vrstva vytváří pakety příslušné síťové architektury. Na přenosové cestě může docházet k poruchám, v jejichž důsledku jsou přijaté bity jiné než odeslané. Fyzická vrstva se nezabývá významem jednotlivých bitů, tento druh chyb rozpozná až linková vrstva. Ta kontroluje, zda byly přeneseny správně celé rámce (posílání různých kontrolních součtů apod.). Odesílateli potvrzuje linková vrstva přijetí bezchybně přenesených rámců, zatímco při chybě si vyžádá jejich opětovné vyslání.

4.3.7. FYZICKÁ VRSTVA (PHYSICAL LAYER)

Tato vrstva popisuje elektrické, mechanické a funkční požadavky na zpracování síťových dat. Úkol má zdánlivě jednoduchý, zajišťuje přenos jednotlivých bitů mezi příjemcem a odesílatelem.

4.4. Model TCP/IP

Zkratka TCP/IP znamená *Transmission Control Protocol/Internet Protocol*, je to standardní soubor protokolů, které se používají např. v síti Internet, která se stala největší počítačovou sítí ve světě.

Z historie víme, že v roce 1969 vyhlásila DARPA (Defense Advanced Projekct Agency) výzkumný a vývojový projekt na vytvoření experimentální sítě s přepínáním paketů. Tato síť se jmenovala ARPANET a vznikla ke studiu technik, poskytujících spolehlivé a na dodavatelích nezávislé datové komunikace. Tento experiment vyšel a řada organizací začala jej využívat k běžné denní datové komunikaci. V roce 1975 se ARPANET změnil z experimentální sítě na operační síť. Základy protokolu TCP/IP vznikly v době, kdy už byl ARPANET operační sítí. Tento protokol byl v roce 1983 přijat jako Military Standard a přibližně v této době se rozšířilo slovo Internet.

Výhody TCP/IP

- Otevřený protokolový standart, volně dostupný a vyvinutý nezávisle na konkrétním technickém vybavení nebo operačním systému.
- Nezávislost na konkrétním fyzickém síťovém hardware. Díky tomu je možné provozovat TCP/IP na řadě různých sítí. TCP/IP může běžet na Ethernetu, token

ringu, telefonní lince a prakticky na každém fyzickém přenosovém médiu.

- Obecné adresové schéma, které umožňuje, aby jakékoliv TCP/IP zařízení jednoznačně adresovalo jakékoliv zařízení v celé síti.
- Standardizované vysokoúrovňové protokoly, které poskytují široce dostupné uživatelské služby.

4.5. Protokolová architektura TCP/IP

Protokoly představují formální pravidla chování v datových komunikacích. V homogenních sítích definuje komunikační pravidla dodavatel systému. TCP/IP vytváří heterogenní síť s otevřenými protokoly, které nezávisí na rozdílech mezi operačními systémy a architekturami. Jsou dostupné pro každého a jejich vývoj a modifikace se řídí dohodou. Nejvíce informací o TCP/IP je publikováno jako RFC (Requests for Comments), tyto dokumenty obsahují poslední verze specifikací všech standardů.

Obvykle se TCP/IP považuje za model složený z méně vrstev než model OSI. Většina popisů TCP/IP definuje protokolovou architekturu pomocí tří až pěti funkčních úrovní.

Vrstva síťového rozhraní (Network Interface Layer)

Tato vrstva má na starosti vše, co je spojeno s ovládáním konkrétní přenosové cesty a s přímým vysíláním a příjmem datových paketů. Je závislá na konkrétní přenosové technologii. *(prakticky všechny průmyslové standardy - Ethernet, IEEE 802.x, FDDI).*

Síťová vrstva (Internet Layer)

Někdy je nazývána také jako IP vrstva, vzhledem k protokolu IP, tato vrstva se stará, aby jednotlivé pakety se dostaly od odesílatele až ke svému skutečnému příjemci, bez ohledu na to, zda mezi nimi existuje přímé spojení nebo ne. Vzhledem k nespojovanému charakteru přenosu (protokol IP) je na úrovni této vrstvy zajišťována jednoduchá datagramová služba.

Internetový protokol je základním kamenem Internetu. Jeho funkce zahrnují:

- definici datagramu, což je základní přenášená jednotka v Internetu
- definici internetového adresovacího schématu
- přenos dat mezi síťovou vrstvou a transportní
- směrování datagramů na vzdálené cíle
- zajištění fragmentace a složení datagramů

Datagram je formát paketu (paket - je blok dat, který obsahuje informace nezbytné pro své doručení) definovaný internetovým protokolem.

Transportní vrstva (Transport layer)

Často též TCP vrstva podle protokolu. Hlavním úkolem této vrstvy je zajistit přenos mezi koncovými účastníky, kterými jsou v případě architektury TCP/IP aplikační programy. Podle jejich požadavků může transportní vrstva regulovat tok dat oběma směry,

zajišťovat spolehlivost přenosu a také měnit nespojový charakter přenosu v síťové vrstvě na spojový.

Aplikační vrstva (Aplication Layer)

Nejvyšší vrstvou architektury TCP/IP je aplikační vrstva, jejími entitami jsou jednotlivé aplikační programy. Ty, na rozdíl od RM OSI, komunikují přímo s transportní vrstvou. Případné prezentační a relační služby si aplikační programy zajišťují samy.

4.6. Architektury komunikujících systémů

Definice pojmů

- **Systém**
  - samostatný celek schopný vykonávat zpracování a přenos informace
- **Síťová architektura**
  - systém vrstev, služeb, funkcí a protokolů,
  - odpovídá struktuře síťového vybavení.
- **Otevřená architektura**
  - obsah norem popisujících architekturu je veřejně přístupný,
  - všechna zařízení (systémy) vyhovující normám jsou vzájemně propojitelná.
- **Vrstvový protokol**
  - pravidla spolupráce entit ve stejné vrstvě na jiných systémech.
- **Mezivrstvový protokol**
  - SW rozhraní,
  - pravidla spolupráce sousedních vrstev,
  - používají se služební primitiva,
  - komunikace prostřednictvím přístupových bodů služeb, (Service Access Point, SAP).

5. Model ISO/OSI, vybrané protokoly

5.1. Počítačové sítě - Model ISO/OSI

Model ISO/OSI je referenční komunikační model označený zkratkou slovního spojení "International Standards Organization / Open Systen Interconnection" (Mezinárodní organizace pro normalizaci / propojení otevřených systémů). Jedná se o doporučený model definovaný organizací ISO v roce 1983, který rozděluje vzájemnou komunikaci mezi počítači do sedmi souvisejících vrstev. Zmíněné vrstvy jsou též známé pod označením *Sada vrstev protokolu*.

Úkolem každé vrstvy je poskytovat služby následující vyšší vrstvě a nezatěžovat vyšší vrstvu detaily o tom jak je služba ve skutečnosti realizována. Než se data přesunou z jedné vrstvy do druhé, rozdělí se do paketů. V každé vrstvě se pak k paketu přidávají další doplňkové informace (formátování, adresa), které jsou nezbytné pro úspěšný přenos po síti.

Uvedený model obsahuje následující vrstvy (každá vyšší vrstva využívá funkce vrstvy nižší).

Nejprve zde stručný popis jednotlivých vrstev, na úvod.

1. Fyzická vrstva

Definuje prostředky pro komunikaci s přenosovým médiem a s technickými prostředky rozhraní. Dále definuje fyzické, elektrické, mechanické a funkční parametry týkající se fyzického propojení jednotlivých zařízení. Je hardwarová.

2. Linková vrstva

Zajišťuje integritu toku dat z jednoho uzlu sítě na druhý. V rámci této činnosti je prováděna synchronizace bloků dat a řízení jejich toku. Je hardwarová.

3. Síťová vrstva

Definuje protokoly pro směrování dat, jejichž prostřednictvím je zajištěn přenos informací do požadovaného cílového uzlu. V lokální síti vůbec nemusí být, pokud se nepoužívá směrování. Je hardwarová, ale když směrování řeší PC s dvěma síťovými kartami, je softwarová.

4. Transportní vrstva

Definuje protokoly pro strukturované zprávy a zabezpečuje bezchybnost přenosu (provádí některé chybové kontroly). Řeší například rozdělení souboru na pakety a potvrzování. Je softwarová.

5. Relační vrstva

Koordinuje komunikace a udržuje relaci tak dlouho, dokud je potřebná. Dále zajišťuje zabezpečovací, přihlašovací a správní funkce. Je softwarová.

6. Prezentační vrstva

Specifikuje způsob, jakým jsou data formátována, prezentována, transformována a kódována. Řeší například háčky a čárky, CRC, kompresi a dekompresi, šifrování dat. Je softwarová.

7. Aplikační vrstva

Je to v modelu vrstva nejvyšší. Definuje způsob, jakým komunikují se sítí aplikace, například databázové systémy, elektronická pošta nebo programy pro emulaci terminálů. Používá služby nižších vrstev a díky tomu je izolována od problémů síťových technických prostředků. Je softwarová.

5.2. Referenční model ISO/OSI - sedm vrstev

Tvůrci referenčního modelu ISO/OSI dospěli k závěru, že optimální počet vrstev síťového softwaru bude sedm. Jaké vrstvy to jsou a jaké úkoly mají řešit? Vezměme to postupně zdola nahoru, od nejnižší vrstvy k nejvyšší:

1. Fyzická vrstva (Physical Layer)

Úkol této vrstvy je zdánlivě velmi jednoduchý - zajistit přenos jednotlivých bitů mezi příjemcem a odesilatelem prostřednictvím fyzické přenosové cesty, kterou tato vrstva bezprostředně ovládá. K tomu je ovšem třeba vyřešit mnoho otázek převážně technického charakteru - např. jakou úrovní napětí bude reprezentována logická jednička a jakou logická nula, jak dlouho "trvá" jeden bit, kolik kontaktů a jaký tvar mají mít konektory kabelů, jaké signály jsou těmito kabely přenášeny, jaký je jejich význam, časový průběh apod. Problematika fyzické vrstvy proto spadá spíše do působnosti elektroinženýrů a techniků.

2. Linková vrstva (Data Link Layer)

Fyzická vrstva poskytuje jako své služby prostředky pro přenos jednotlivých bitů. Bezprostředně vyšší linková vrstva (někdy nazývaná též: **spojová vrstva** či **vrstva datového spoje**) pak má za úkol zajistit pomocí těchto služeb bezchybný přenos celých bloků dat (velikosti řádově stovek bytů), označovaných jako **rámce** (**frames**). Jelikož fyzická vrstva nijak neinterpretuje jednotlivé přenášené bity, je na linkové vrstvě, aby správně rozpoznala začátek a konec rámce, i jeho jednotlivé části.

Na přenosové cestě může docházet k nejrůznějším poruchám a rušení, v jejichž důsledku jsou přijaty jiné hodnoty bitů, než jaké byly původně vyslány. Jelikož fyzická vrstva se nezabývá významem jednotlivých bitů, rozpozná tento druh chyb až linková vrstva. Ta kontroluje celé rámce, zda byly přeneseny správně (podle různých kontrolních součtů, viz 3. díl našeho seriálu). Odesilateli potvrzuje přijetí bezchybně přenesených rámců, zatímco v případě poškozených rámců si vyžádá jejich opětovné vyslání.

3. Síťová vrstva (Network Layer)

Linková vrstva zajišťuje přenos celých rámců, ovšem pouze mezi dvěma uzly, mezi kterými vede přímé spojení. Co ale dělat, když spojení mezi příjemcem a odesilatelem není přímé, ale vede přes jeden či více mezilehlých uzlů? Pak musí nastoupit síťová

vrstva, která zajistí potřebné směrování (routing) přenášených rámců, označovaných nyní již jako pakety (packets). Síťová vrstva tedy zajišťuje volbu vhodné trasy resp. cesty (route) přes mezilehlé uzly, a také postupné předávání jednotlivých paketů po této trase od původního odesilatele až ke konečnému příjemci.

Síťová vrstva si tedy musí "uvědomovat" konkrétní topologii sítě (tj. způsob vzájemného přímého propojení jednotlivých uzlů).

4. Transportní vrstva (Transport Layer)

Síťová vrstva poskytuje bezprostředně vyšší vrstvě služby, zajišťující přenos paketů mezi libovolnými dvěma uzly sítě. Transportní vrstvu proto zcela odstiňuje od skutečné topologie sítě a vytváří jí tak iluzi, že každý uzel sítě má přímé spojení s kterýmkoli jiným uzlem sítě.

Transportní vrstvě díky tomu stačí zabývat se již jen komunikací koncových účastníků (tzv. end-to-end komunikací) - tedy komunikací mezi původním odesilatelem a konečným příjemcem.

Při odesílání dat zajišťuje transportní vrstva sestavování jednotlivých paketů, do kterých rozděluje přenášená data, a při příjmu je zase z paketů vyjímá a skládá do původního tvaru. Dokáže tak zajistit přenos libovolně velkých zpráv, přestože jednotlivé pakety mají omezenou velikost.

5. Relační vrstva (Session Layer)

Úkolem této vrstvy je navazování, udržování a rušení **relací** (**sessions**) mezi koncovými účastníky. V rámci navazování relace si tato vrstva vyžádá na transportní vrstvě vytvoření spojení, prostřednictvím kterého pak probíhá komunikace mezi oběma účastníky relace. Pokud je třeba tuto komunikaci nějak řídit (např. určovat, kdo má kdy vysílat, nemohou-li to dělat oba účastníci současně), zajišťuje to právě tato vrstva, která má také na starosti vše, co je potřeba k ukončení relace a zrušení existujícího spojení.

6. Prezentační vrstva (Presentation Layer)

Data, která se prostřednictvím sítě přenáší, mohou mít mj. povahu textů, čísel či obecnějších datových struktur. Jednotlivé uzlové počítače však mohou používat odlišnou vnitřní reprezentaci těchto dat - např. střediskové počítače firmy IBM používají znakový kód EBCDIC, zatímco většina ostatních pracuje s kódem ASCII. Podobně jeden počítač může zobrazovat celá čísla v doplňkovém kódu, zatímco jiný počítač v přímém kódu apod. - potřebné konverze přenášených dat má na starosti právě tato prezentační vrstva.

V rámci této vrstvy bývá také realizována případná komprese přenášených dat, eventuálně i jejich šifrování.

7. Aplikační vrstva (Application Layer)

Koncoví uživatelé využívají počítačové sítě prostřednictvím nejrůznějších síťových aplikací - systémů elektronické pošty, přenosu souborů, vzdáleného přihlašování (remote login) apod. Začleňovat všechny tyto různorodé aplikace přímo do aplikační vrstvy by pro jejich velkou různorodost nebylo rozumné. Proto se do aplikační vrstvy

zahrnují jen části těchto aplikací, které realizují společné resp. obecně použitelné mechanismy. Uvažujme jako příklad právě elektronickou poštu - ta její část, která zajišťuje vlastní předávání zpráv v síti, je součástí aplikační vrstvy. Na všech uzlových počítačích, které používají tentýž systém elektronické pošty, je tato část stejná. Uživatelské rozhraní systému elektronické pošty, tedy ta jeho část, se kterou uživatel bezprostředně pracuje a jejímž prostřednictvím čte došlé zprávy, odpovídá na ně, připravuje nové zprávy a zadává je k odeslání, již není považována za součást aplikační vrstvy, neboť se může v každém konkrétním uzlu dosti výrazně lišit - např. ve způsobu svého ovládání (řádkovými příkazy či pomocí různých menu, s okénky či bez nich apod.). Jiným názorným příkladem může být emulace terminálů, potřebná např. pro vzdálené přihlašování (remote login). Ve světě dnes existuje nepřeberné množství různých terminálů, a realizovat potřebné přizpůsobení mezi libovolnými dvěma typy terminálů je v podstatě nemožné. Proto se zavádí jediný "referenční" terminál - tzv. virtuální terminál - a pro každý konkrétní typ terminálu se pak vytvoří jen jediné přizpůsobení mezi tímto virtuálním terminálem a terminálem skutečným. Prostředky pro práci s virtuálním terminálem přitom jsou součástí aplikační vrstvy (neboť jsou všude stejné), zatímco prostředky pro jeho přizpůsobení konkrétnímu terminálu již součástí aplikační vrstvy nejsou.

6. Bezdrátové komunikační technologie

Bezdrátové sítě. Princip, standard, provedení prvků, režimy komunikace, použití a vlastnosti.

Standard

Vývoj bezdrátových sítí probíhal podobně jako u sítí kabelových. Nejdříve živelně, posléze bylo nutné přijmout normu, která zajistí vzájemnou spolupráci sítí. Hlavní výrobci bezdrátové technologie založili alianci WECA (Wireless Ethernet Compatibility Alliance – sdružení pro kompatibilitu bezdrátového Ethernetu), která stanovila požadavky na zařízení a zajistila tak vzájemnou kompatibilitu. Při splnění podmínek obdrží výrobek certifikát Wi-Fi, který potvrzuje kompatibilitu s výrobky ostatních výrobců. Samotná wireless norma byla odvozena z Ethernetu, proto s ním má některé podobné znaky – přístupovou metodu CSMA/CD a obdobné složení paketu. Pro bezdrátové sítě LAN existuje několik standardů, jejichž základní vlastnosti vidíte v tabulce. Standard 801.11g je zpětně kompatibilní se starším a pomalejším 802.11b (mohou spolupracovat, samozřejmě na nižší rychlosti). Ze standardu 802.11g byla odvozena norma 802.11i, používající bezpečnější autentizační a šifrovací algoritmus.

Provedení prvků

Bezdrátové prvky spolu mohou komunikovat dvěma způsoby:

- ad hoc, kdy se jedná vlastně o přímé propojení několika počítačů – od dvou do pěti. Každý počítač komunikuje s jiným na stejné úrovni, jsou si rovni (organizace je podobná síti peer to peer). Podstatnou výhodou je rychlá instalace a velmi nízká cena (kromě klientských síťových adaptérů nepotřebujeme žádný další hardware). Umožňuje sdílení souborů a Internetu, tisk přes síť a ostatní věci, které jsou běžné u klasických sítí LAN. Nevýhodouje fakt, že všechna připojená zařízení musí být v dosahu – každý musí vidět každého. Dalším nedostatkem je to, že spojení vzniká až nebezpečně snadno a je obtížné je zabezpečit.
- infrastrukturní mód založený na přístupovém bodu – Access Pointu (AP). Ten pracuje jako prostředník (server), přes nějž proudí všechny datové toky mezi klienty sítě (organizace je podobná síti klient/server). Jeho použití má především výhodu v možnosti filtrovat či kontrolovat provoz, včetně zpřístupnění sítě různým klientům. To zaručuje eliminaci náhodných pokusů o sestavení ad hoc spojení, veškerý tok musí směřovat na AP, což umožňuje síť ochránit. Přístupový bod není rozhodně nutný, pokud provádíme bezdrátové spojení příležitostně, mezi několika zařízeními. Budujete-li však malou domácí síť s více účastníky, či hodláte sdílet třeba Internet domácnosti či malé kanceláři, bez přístupového bodu se většinou neobejdete (vyšší zabezpečení sítě).

Přístupový bod – AP

Je základem bezdrátové sítě. Zprostředkovává spojení mezi bezdrátovými koncovými

body a serverem, většinou umístěným v metalické síti LAN. AP tedy obsahuje radiovou část – vysílač/přijímač a část kabelovou – zdířky RJ-45 pro připojení kroucené dvojlinky. AP jsou huby, z nichž se rozvádí signál. Mnoho výrobců nabízí napájení AP bodu pomocí kroucené dvojlinky, jíž je bod připojen k pevné síti. K přístupovému bodu (na obtížně dostupném místě) se tak nemusí táhnout dvě vedení. Přístupový bod a jeho protějšky – klientské adaptéry pracují pouze tehdy, pokud mezi nimi není žádná překážka – mezi AP a počítači umístěnými v bezdrátové síti musí být přímá viditelnost. Proto najdeme přístupové body v nejvyšších částech místností. Při jejich umístění musíme brát do úvahy možné zdroje rušení rádiového signálu, kovové konstrukce (i ve zdi), elektrická rušení (v pásmu 2,4 GHz pracují např. mikrovlnné trouby, bezšňůrové telefony, bezdrátové reproduktory). Pokud je potřeba propojit bezdrátové sítě mezi sebou, je možné použít Wireless Bridge (WB) – přístupové body s funkcí mostů, pro filtrování paketů mezi sítěmi.

Klientský adaptér

Jde o jednotku, jíž je „připojeno“ PC k přístupovému bodu. V podstatě to tedy je síťová karta (s anténkou). Její provedení bývá pro sloty PCI či USB. Do notebooků lze použít síťovou wireless kartu normy PC Card, ale mnoho notebooků má bezdrátové síťové rozhraní již integrováno (což dále zjednodušuje a zlevňuje budování bezdrátové sítě).

6.1. Vlastnosti bezdrátové sítě

6.1.1. RYCHLOST

Je ve srovnání s metalickými sítěmi podstatně nižší. V tabulce, popisující základní normy, jsou uvedeny maximální teoreticky dosažitelné rychlosti. Ty jsou ve skutečnosti těžko dosažitelné, protože je horší dostupnosti signálu mezi rádiovými stanicemi dojde k přeskoku na nižší přenosovou rychlost.

V praxi to znamená, že při větší vzdálenosti (třeba přes 20 m) či při zastínění (stačí kvalitní kovová zárubeň) okamžitě rychlost klesá, a to rapidně, třeba o polovinu či čtvrtinu. Pokud je skupina příjemců připojena na stejný přístupový bod a nacházejí se tak fyzicky na jednom síťovém segmentu, musejí se o kapacitu linky podělit – rozbočovačem. Dochází tak ke kolizím (znak CSMA/CD). Dalším činitelem snižujícím reálnou rychlost je nutná režie protokolů vyšších vrstev. Skutečná šíře pásma, určená pro čistá data, tak opět povážlivě klesá. Za velmi dobrých podmínek se maximální šířka pásma pro užitečný datový náklad pohybuje někde kolem poloviny nominálních hodnot, tedy asi 5 Mb/s u „béčka“ a 25 Mb/s u „géčka“.

6.1.2. BEZPEČNOST

SSID *(Service Set ID)*

Je názvem Access Pointu, pod nímž jej uvidí všichni klienti, kteří se dostanou do jeho

dosahu. SSID je tak logickým identifikátorem určité bezdrátové podsítě. Může být nastaven manuálně na stanici, nebo informaci o SSID přístupový bod pravidelně vysílá, či může být vysílání SSID vypnuto a klient se na SSID sám dotáže.

WEP (Wired Equivalent Privacy)

Zastaralé zabezpečení bezdrátových sítí podle původního standardu IEEE 802.11. Cílem WEP bylo poskytnout zabezpečení obdobné drátovým počítačovým sítím (např. kroucená dvojlinka), protože rádiový signál je možné snadno odposlouchávat i na delší vzdálenost bez nutnosti fyzického kontaktu s počítačovou sítí. WEP byl prolomen v srpnu 2001, a proto by jeho nasazení mělo být nahrazeno zabezpečením pomocí WPA2 podle standardu IEEE 802.11i.

WPA (Wi-Fi Protected Access)

Označení pro zabezpečení bezdrátových sítí. Po prolomení zabezpečení WEP v roce 2001 definovala Wi-Fi Alliance v roce 2002 zabezpečení WPA pro Wi-Fi sítě jako část tehdy připravovaného standardu IEEE 802.11i. Wi-Fi Alliance vlastní práva na značku Wi-Fi i WPA a certifikuje zařízení, která ji nesou.

WPA2

Implementuje všechny povinné prvky IEEE 802.11i. Používá blokovou šifru Advanced Encryption Standard (AES), zatímco dřívější WEP a WPA používají proudovou šifru RC4. 802.11i architektura obsahuje následující komponenty: IEEE 802.1X pro autentizaci (používá tedy Extensible Authentication Protocol (EAP).

Wireless PAN

(WPAN) spojují jednotlivá zařízení v relativně malé oblasti. Která je obecně pro osobu připojenou do této sítě snadno dosažitelná. Například pomocí bluetooth nebo infračerveného světla můžeme připojit sluchátka k laptopu a tím si vytvořit malou osobní bezdrátovou síť (WPAN). ZigBee také podporuje WPAN aplikace. Osobní Wi-Fi sítě se staly samozřejmostí jako vybavení integrované do celé škály elektronických zařízení pro běžné spotřebitele. Nástroje „My WiFi“ od Intelu a „Virtual Wi-Fi“ z Windows 7 umožňují osobní bezdrátové sítě snadno a jednoduše nastavit a konfigurovat.

Wireless WWAN

Velká bezdrátová síť (WWAN) je bezdrátová síť, která typicky pokrývá velké oblasti. Tyto sítě mohou být použity k připojení poboček kanceláří nebo jako veřejný přístupový systém. Bezdrátové spojení mezi přístupovými body je obvykle point-to-point mikrovlnná linka používající parabolický reflektor na frekvenci 2,4 GHz. Typický systém obsahuje vstupní brány základních stanic, přístupové body a bezdrátové přemostění signálu. Ostatní konfigurace jsou síťové systémy, kde každý přístupový bod předává signál dál.

Wireless MAN

Bezdrátové metropolitní sítě (WMAN) jsou typ bezdrátové sítě, které spojuje několik

bezdrátových lokálních sítí. WiMAX je typ bezdrátové MAN a je popsána standardem IEEE 802.16

6.1.3. POUŽITÍ

Zahrnuje mobilní telefony, které jsou částí každodenní bezdrátové komunikace, umožňující snadnou komunikaci. Dalším příkladem je mezikontinentální síťový systém, který používá satelity ke komunikaci napříč celým světem. Běžní lidé a obchodníci využívají bezdrátové sítě k rychlému posílání a sdílení dat, ať už jsou v malé kanceláři, nebo kdekoli na světě.

7. Virtuální sítě (VLAN).

VLAN je takzvaná virtuální lokální síť (z anglického *Virtual Local Area Network*). **V rámci funkční sítě**

LAN je tedy vytvořená další, virtuální síť, která funguje i v rámci jen jednoho hardwarového zařízení (*fyzická kabeláž*). V případě VLAN se také dá říci, že jde o *individuální přenastavení již aktivních síťových prvků do nového virtuálního kabátu* a modelu.

VLAN funguje na principu značkování dat, které se po síti posílají. Data se označují podle virtuální sítě, do které patří. Jde tak o náležité propojení více lokálních LAN na úroveň druhé síťové vrstvy ISO/OSI modelu. Velkou výhodou je následně dobrá konfigurovatelnost.

V rámci dobré úrovně bezpečnosti sítě VLAN umí nezávisle na aktivním síťovém zařízení od sebe inteligentně oddělit uživatele jednotlivých koncových stanic a stále běžící uživatelské aplikace (např. linuxový démon).

Neoprávněný přístup do počítačové sítě ze strany neetického hackera je tak razantně eliminován. Nejčastější typ kybernetického útoku vedeného na počítačové sítě – DDoS, tak může velmi lehce odražen v samém začátku.

7.1. Proč existuje síť VLAN?

První tzv. nástřely VLAN technologie se začaly dělat v polovině 90. let minulého století. Jeden z hlavních důvodů bylo seskupení určitých uživatelů do jedné skupiny, která spolu bude komunikovat a bude mít tak snazší přístup ke svým souborům a informacím. Dalším důvodem bylo stagnace ve vývoji ethernetové technologie.

Důležitým bodem je také to, že technologie VLAN flexibilně odděluje síťovou komunikaci, která se vždy nově přiřazuje do sítě na příslušném síťovém zařízení - switch.

V případě použití portu switche se jedná zřejmě o nejefektivnější řešení v praxi zařazení do sítě.

Technologie VLAN začala vznikat kolem roku 1995, ale zprvu se jednalo o různá proprietární řešení. V praxi se však více rozšířili až před několika lety a to hlavně ve středních a velkých firmách, přestože již delší dobu existuje standard.

Hlavní důvody proč vznikly VLAN byly tyto:

- *seskupování uživatelů* v síti podle skupin či oddělení nebo podle služeb místo podle fyzického umístění a oddělení komunikace mezi těmito skupinami,
- *snížení broadcastů* v síti, které začaly být problémem již před několika lety,
- *zmenšení kolizních domén* v době, kdy se nepoužívaly switche, ale třeba huby.

Idea pro logické seskupování uživatelů, která se uvádí v řadě materiálů, a tedy vytváření

VLAN je

- *podle organizační struktury* - pokud je většina komunikace v rámci oddělení, kde jsou vlastní tiskárny, file servery, atd. a mezi jednotlivými odděleními není komunikace, pouze pár služeb (mail) je společných pro všechny
- *podle služeb* - do VLAN se seskupují pracovníci, kteří využívají stejné služby (účetnictví, DB, atd.)

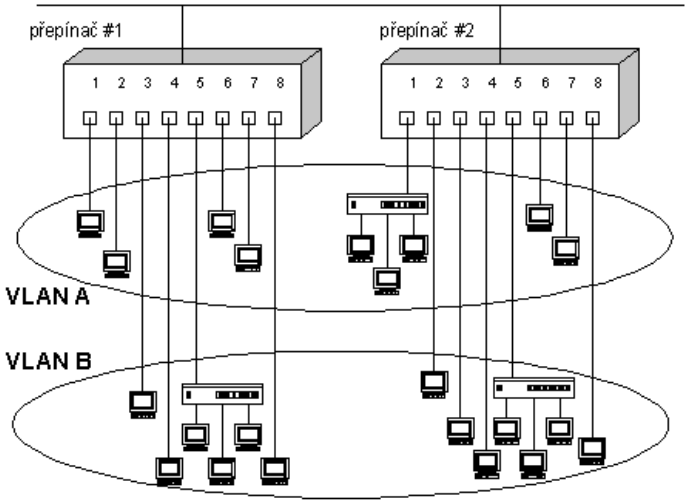
Jaké jsou praktické výhody VLAN

Protože lze členství ve VLAN definovat různými způsoby, typologicky se VLAN rozdělují se na sítě se členstvím podle portů a na tzv. policy-based. Podrobnější praktické dělení pak rozeznává čtyři typy VLAN se členstvím podle:

- portů;
- MAC adres uzlů;
- síťového protokolu nebo síť. adres uzlů;
- skupinového IP vysílání.

VLAN podle portů.

Tento historicky první typ virtuálních sítí definuje členství v síti pro jednotlivé porty přepínače (skupiny portů). První implementace těchto VLAN neumožňovaly rozšíření virtuální sítě přes více přepínačů, byly omezeny pouze na jeden přepínač. Druhá generace to již dovolila, příklad takto definované sítě je na obrázku.



Obr. - VLAN se členstvím podle portů

Seskupování portů je stále nejpoužívanější metodou při vytváření virtuálních sítí. Je sice velmi jednoduchá a názorná, její základní omezení ale spočívá v nutnosti předefinování členství při jakémkoliv přesunu uživatelské stanice mezi jednotlivými porty přepínače (tedy přesněji řečeno takové změně portu, při které by došlo ke změně členství ve virtuální síti).

VLAN podle MAC adresy

MAC adresa je "natvrdo zadrátovaná" v obvodech síťového adaptéru, takže na takto definované virtuální sítě lze pohlížet jako na VLAN podle uživatelů. Změní-li totiž uživatel svoje připojení (přemístí se svojí stanicí na jiný segment/port přepínače), jeho členství ve VLAN se nezmění.

Při této metodě je nutností prvotní manuální definice členství pro všechny stanice sítě. Jinou nevýhodou je možnost podstatného snížení výkonu v případě, že na portu přepínače je připojen sdílený segment se stanicemi v různých VLAN. Dalším, spíše ale jen okrajovým problémem může být situace, kdy uživatelé s mobilními notebooky mění svoji pozici a připojují se pomocí stabilně připojených docking stations. Tím se jím definiční MAC adresa s lokalitou mění (síťový adaptér je většinou součástí docku). Ačkoliv je to minoritní problém, dobře ilustruje jistá omezení VLAN, založených na členství podle MAC adresy.

Možnost uživatelského předefinování vlastní MAC adresy přímo v operačních systémech tento problém např. z bezpečnostního hlediska ještě více komplikuje.

VLAN podle protokolu nebo adres

Takto definované virtuální sítě jsou založeny na informacích ze třetí, tedy síťové vrstvy podle OSI modelu. V multiprotokolových sítích mohou být přiřazeny uzly do jednotlivých VLAN podle provozovaných síťových protokolů nebo např. v sítích s protokolem TCP/IP podle adresy podsítě.

Ačkoliv se zde pracuje s informacemi síťové vrstvy, je důležité si uvědomit, že se nejedná o jejich využití pro směrování. I když přepínač musí prohlédnout paket k určení IP adresy a tím členství ve VLAN, neprovádí žádné směrovací výpočty. Přepínač nevyužívá směrovací protokoly (jako jsou RIP, OSPF) a i na VLAN definovanou podle informací ze třetí, síťové vrstvy se musíme dívat jako na síť s plochou topologií, propojenou přepínači či můstky.

Rozmach přepínání i na třetí, síťové vrstvě a existence přepínačů se zabudovanými schopnostmi směrovačů tento problém při prvním pohledu trochu zamlžuje. Jedná se ale o odlišné funkce - pouhé určení členství ve VLAN na základě síťové adresy v jednom případě a plné využití směrovacích schopností na základě směrovacích protokolů a výpočtů na straně druhé. Zde je nutné podotknout, že komunikace mezi jednotlivými VLAN vyžaduje použití směrovačů - ať už klasických nebo právě těchto zabudovaných v přepínačích se směrovacími funkcemi.

Způsob definice VLAN podle síťové vrstvy má své zřejmé výhody. Patří mezi ně:

- mobilita uživatelů či přesněji řečeno jejich stanic bez nutnosti překonfigurování členství ve VLAN,
- možnost vytváření skupin specifických pro jistou službu nebo aplikaci,
- eliminace potřeby značkování paketů informací o členství ve VLAN při vzájemné komunikaci přepínačů (viz dále).

VLAN podle skupinového vysílání

Skupinové vysílání v IP sítích (IP multicast) pracuje tak, že paket, určený ke skupinovému

vysílání je poslán na speciální adresu, která funguje jako proxy pro explicitně definovanou skupinu uzlů (IP adres). Paket je pak doručen všem uzlům, které jsou členy dané skupiny. Ta se sestavuje dynamicky, uzly se do této skupiny průběžně přihlašují a odhlašují.

Na všechny stanice takovéto skupiny můžeme tedy pohlížet jako na členy jedné virtuální LAN, protože skupina tvoří jednu doménu všesměrového vysílání. Od předchozích uvedených typů se liší ve dvou podstatných rysech - je vytvářena dynamicky jen na určitou dobu, takže je velmi flexibilní a její rozsah není omezen směrovači, tzn., že se může rozprostírat např. i po rozlehlé síti WAN.

Proč vznikly VLANy

Technologie VLAN začala vznikat kolem roku 1995, ale zprvu se jednalo o různá proprietární řešení. V praxi se však více rozšířili až před několika lety a to hlavně ve středních a velkých firmách, přestože již delší dobu existuje standard.

Hlavní důvody proč vznikly VLAN byly asi tyto:

- seskupování uživatelů v síti podle skupin či oddělení nebo podle služeb místo podle fyzického umístění a oddělení komunikace mezi těmito skupinami,
- snížení broadcastů v síti, které začaly být problémem již před několika lety,
- zmenšení kolizních domén v době, kdy se nepoužívaly switche, ale třeba huby,
- Idea pro logické seskupování uživatelů, která se uvádí v řadě materiálů, a tedy vytváření VLAN je,
- podle organizační struktury - pokud je většina komunikace v rámci oddělení, kde jsou vlastní tiskárny, file servery, atd. a mezi jednotlivými odděleními není komunikace, pouze pár služeb (mail) je společných pro všechny,
- podle služeb - do VLAN se seskupují pracovníci, kteří využívají stejné služby (účetnictví, DB, atd.).

7.2. Jaké jsou praktické výhody VLAN

Jak se zařazuje komunikace do VLAN

Přiřazení do VLANy se nastavuje typicky na switchi (pouze v některých speciálních případech přichází označená komunikace přes trunk z jiného zařízení). Na switchech, které podporují VLANy, vždy existuje alespoň jedna VLAN. Jedná se o defaultní VLAN číslo 1, kterou není možno smazat či vypnout. Pokud nenastavíme jinak, tak jsou všechny porty (tedy veškerá komunikace) zařazeny do VLAN 1.

Pro zařazení komunikace do VLANy existují čtyři základní metody, ale v praxi je nejvíce využívána možnost první - zařazení dle portu.

1. podle portu

Port switche je ručně a napevno zařazen (nakonfigurován) do určité VLANy. Veškerá komunikace, která přichází přes tento port, spadá do zadané VLANy. To znamená, že

pokud do portu připojíme další switch, tak všechny zařízení připojená k němu budou v jedné VLANě. Jedná se o nejrychlejší a nejpoužívanější řešení. Není třeba nic vyhodnocovat pro zařazení do VLAN. Definice zařazení do VLAN je lokální na každém switchi. Jednoduše se spravuje a je přehledné.

2. podle MAC adresy

Rámce(port) se zařadí do VLANy podle zdrojové MAC adresy. Musíme tedy spravovat tabulku se seznamem MAC adres pro každé zařízení spolu s VLANou. Výhodou je, že se jedná o dynamické zařazení, takže pokud přepojíme zařízení do jiného portu, automaticky se zařadí do správné VLANy. Switch musí vyhledávat v tabulce MAC adres.

Jsou zde dvě možnosti, jak tato metoda může fungovat. Buď se podle MAC adresy prvního rámce nastaví zařazení portu do VLANy a toto nastavení zůstane, dokud se port nevypne. Nebo se každý rámec zařazuje samostatně do VLANy podle MAC adresy. Toto řešení je velmi náročné na výkon.

Cisco má řešení zvané VLAN Membership Policy Server (VMPS), pro které je třeba speciální server, který spravuje tabulky MAC adres. Navíc se při této metodě zařazuje port do VLANy, takže pokud je do něj připojeno více zařízení (max. 20), musí být všechny ve stejné VLAN.

3. podle protokolu = podle informace z 3. vrstvy

Tato metoda určuje zařazení podle protokolu přenášeného paketu. Například oddělíme IP provoz od AppleTalk. Nebo zařazujeme podle IP adresy či rozsahu. V praxi není příliš rozšířené. Zařízení musí mít napevno definovanou IP adresu a switch se musí dívat do třetí vrstvy (normálně funguje na druhé), znamená to zpomalení.

4. podle autentizace

Ověří se uživatel nebo zařízení pomocí protokolu IEEE 802.1x a podle informací se automaticky umístí do VLANy. Je to primárně bezpečnostní metoda, které řídí přístup do sítě (NAC), ale po rozšíření slouží i pro VLANy. Je to zajímavá metoda proto, že je velmi univerzální. Nezáleží ani na fyzickém zařízení ani na místě zapojení. RADIUS server, který ověřuje identitu uživatele, obsahuje také mapovaní uživatelů na VLANy a tuto informaci zašle po úspěšné autentizaci. U této metody je možné nastavení, že v případě, kdy není uživatel autentizován, tak je zařazen do speciální hostovské VLANy.

U Cisco switchů může být port single-host, kdy je možno připojit pouze jedno zařízení nebo multiple-host, kdy sice může být do portu připojeno více zařízení, ale ve chvíli, kdy se první autentizuje, tak je port autentizovaný (a zařazený do VLANy) a komunikovat mohou všechna zařízení.

7.3. Jak funguje komunikace v rámci VLAN

V praxi máme dvě situace, kdy se při komunikaci řeší příslušnost k VLANě. Je to při komunikaci v rámci jednoho switche nebo při komunikaci mezi několika switchi.

VLANy na jednom switchi

Při komunikaci ve VLANách v rámci jednoho switche je to jednoduché. Switch si v operační paměti udržuje informace, do které VLANy patří daná komunikace (port), a v rámci switche povoluje pouze správné směrování. V tomto případě máme jednotlivé porty zařazeny do jedné VLANy a to buď staticky, nebo dynamicky, jak bylo řečeno výše (možnosti 2,3,4). Cisco těmto portům říká access port (přístupový port).

VLANY mezi více switchi

Složitější situace nastává, když chceme, aby se informace o zařazení do VLANy neztratila při přechodu na jiný switch, tedy abychom v celé naší síti mohli využít stejné VLANy a nezáleželo, do kterého switche je zařízení připojeno. Navíc chceme, aby tato metoda fungovala i mezi switchi různých výrobců. To byl ze začátku problém a používali se různé metody. Například, když zařazujeme komunikaci podle MAC adresy, tak můžeme tabulku přiřazení mít na všech switchích. Cisco vytvořilo svoji metodu ISL, která zapouzdřuje celý rámec, ale funguje pouze na Cisco zařízeních. Také můžeme propojit dva access porty na dvou switchích, zařadit je do stejné VLANy a přeneseme potřebné informace. To je ale velmi nepraktické.

8. URL, X/HTML, HTTP

URL je zkratka z anglického *Uniform Resource Locator*. Používá se pro přesnou identifikaci dokumentů na internetu. Příklad URL stránky je

http://www.adaptic.cz/znalosti/slovnicek/url/

kde můžete rozeznat adresu našeho serveru skládající se z domény nejvyššího řádu (cz), domény druhé úrovně (adaptic) a domény třetí úrovně (www) oddělených od sebe tečkami. URL dále obsahuje cestu ke stránce v rámci struktury adresářů (/znalosti/slovnicek/url/) oddělovaných lomítky. Poslední částí URL je protokol, přes který je možné o tuto stránku server požádat, v tomto případě jde o protokol HTTP (ono http:// na začátku).

URL může dále obsahovat:

- název stránky včetně její koncovky, (třeba index.htm),
- číslo portu, který identifikuje požadovanou službu (píše se za TLD a odděluje se dvojtečkou, třeba :80),
- jméno a heslo, pokud to server vyžaduje (píše se hned za protokol způsobem uzivatelske-jmeno:heslo@),
- součástí URL mohou být také záložky ukazující na určité místo na stránce (zapisují se jako #zalozka na úplný konec URL),
- parametry stránky (píší se za jménem stránky a oddělují se otazníkem, například ?logged=true).

8.1. Druhy URL

Formě URL zmíněné v příkladu výše říkáme **absolutní URL**. Proti tomu existují **relativní URL**, jež vychází z umístění aktuálního dokumentu (stránky, která odkaz obsahuje). V takovém případě je možné některé části vynechat, např. pokud je odkazovaná stránka umístěna na témže serveru, můžeme v URL vynechat jméno serveru (domény). Pokud je navíc umístěna ve stejném adresáři jako odkazující stránka, není třeba v URL psát ani cestu.

Dobře ošetřeným URL se anglicky říká **cool URL** (přešlo i do češtiny). Zajímá-li nás jen použitelnost, pak mluvíme o **přátelských URL** (někdy také *user friendly URL*). Pokud naopak hledíme pouze na vyhledávače, pak je řeč o **SEOfriendly URL**.

8.2. Proč je podoba URL důležitá

Podoba URL přímo ovlivňuje viditelnost a použitelnost webu. Například je-li jméno domény krátké a srozumitelné, budou si ho návštěvníci spíše pamatovat a také je možné jej snáze nadiktovat po telefonu. Obsahuje-li jméno domény (nebo některé jiné

části URL) klíčové slovo, je to užitečné z hlediska optimalizace pro vyhledávače. Opačným způsobem v obou případech fungují URL obsahující velké množství parametrů, URL psaná velkými písmeny či příliš dlouhá URL (ta se také zalamují v e-mailech a mailový program je znečitelní).

URL

URL se do stránek zapisuje buďto jako absolutní adresa, nebo jako relativní. Zatímco absolutní adresa je to samé, co URL, relativní adresa je nějaký zkrácený zápis adresy, který funguje díky tomu, že prohlížeč ten zápis pochopí podle adresy aktuální stránky. Pozor, v URL záleží na velikosti písmen!

Skladba absolutní URL

Každá absolutní adresa se skládá z protokolu a doménového jména. Většinou následuje adresářová cesta a jméno souboru. Někdy jsou v adrese ještě číslo portu, jméno záložky a dotaz (query string).

Protokoly

HTML stránky se nejčastěji přenášejí http nebo https protokolem. Ty se do URL zapisují jako http:// a https://

Z pohledu tvorby stránek je rozdíl mezi http a https nepodstatný. Rozdíl nastává až při komunikaci serveru s prohlížečem. Http se internetem přenáší nezakódovaně, https je protokol šifrovaný.

Při psaní adres nelze http a https zaměňovat, protože na serveru nemusejí oba protokoly fungovat. Používáte-li absolutní adresy, zjistěte si, na jakém protokolu daný web běží.

Domény

Každý server na Internetu má doménové jméno. To se skládá ze tří částí oddělených tečkami.

- jméno virtuálního serveru, nejčastěji www (doména třetí úrovně)
- jméno domény druhé úrovně (nutná registrace)
- doména nejvyššího řádu, nejčastěji cz, sk nebo com

Port

Velmi zřídka se za generickou doménu píše dvojtečka a číslo portu.

Adresářová cesta

Cílový soubor bývá uložen v adresáři (ale může být přímo v kořeni serveru). Adresáře se do URL zapisují za generickou doménu. Před jméno adresáře patří lomítko (obyčejné, nikoli zpětné). Je-li adresářů více úrovní, píšou se za sebe odděleny lomítky.

Soubory

Vlastní jméno souboru se píše za adresářovou cestu (existuje-li). Před jméno souboru se dává lomítko.

Záložky

Odkaz může mířit na záložku v odkazovaném dokumentu. Do URL se za jméno souboru píše křížek # a jméno záložky.

Dotaz

Součástí URL mohou být i vstupní data pro nějaký skript. Ta se do URL píšou na konec za otazník. Syntaxe je

```
jmeno=hodnota&jmeno2=hodnota2.
```

Příklad

Příkladem absolutního URL může být:
http://www.jakpsatweb.cz/html/url.htm#priklad

Část adresy	Příklad	Jiné možné hodnoty
protokol	http://	ftp:// , mailto: atd.
doména 3. úrovně (server)	www.	www. , cokoliv.
doména 2. úrovně	jakpsatweb.	seznam. , mujweb. apod.
doména nejvyššího řádu	cz	com, sk, gov apod.
port	nic	:80 , :*číslo*
cesta (adresáře)	/html/	/, /cokoliv/adresar/
jméno souboru	url.htm	index.html apod.html
záložka	#priklad	#jménozáložky
dotaz	nic	?proměnná=hodnota

Relativní adresování

Mnohdy je vypisování celé absolutní adresy zbytečné a zdlouhavé. Existuje možnost, jak si práci usnadnit použitím relativní adresy.

Myšlenka relativních adres se zakládá na tom, že soubory, které se navzájem odkazují, často leží na tomtéž serveru. Každý soubor, který pomocí URL vyžaduje jiný soubor, má sám nějaké absolutní URL. Takže stačí, aby se do adresy napsaly cesta k souboru, lomítko a jméno souboru. To je relativní URL.

relativní URL = cesta/jméno_souboru

Často oba soubory leží v jednom adresáři, takže pak není nutno vyplňovat cestu (je prázdná), relativní URL tvoří pouze **jméno souboru**.

Adresáře se oddělují lomítky. Pokud se cílený soubor nachází v hierarchii adresářů někde výše (takže je nutno "vyskákat" nahoru), použije se zápis dvou teček pro nadřazený adresář.

Příklad: Vložení obrázku s logem Jak psát web do této stránky za použití relativní adresy by se udělalo takto: <img src="../images/jakpw.png">

V relativním URL lze samozřejmě použít záložky a dotazy.

8.3. XHTML

XHTML je moderní značkovací jazyk sloužící jako nástupce dnes již zastaralého jazyka HTML. XHTML je zároveň aplikací jazyka XML, z čehož plynou některé odlišnosti, například nutnost deklarace kódování, přísnější pravidla pro zápis elementů (např. musí být uzavřené) a atributů (malými písmeny, v uvozovkách).

XHTML dnes existuje ve dvou verzích. První je XHTML 1.0, dělící se ještě na tři další varianty, Frameset (varianta pro stránky používající rámy), Transitional (varianta mající usnadnit přechod na XHTML), a Strict (nejpřísnější varianta). Druhou verzí je XHTML 1.1, které se příliš neliší od varianty XHTML 1.0 Strict, pouze je rozděleno do několika modulů.

Ačkoliv na tuto problematiku existuje množství různorodých názorů, v praxi se nejlépe osvědčuje používat k tvorbě www stránek právě striktní XHTML. XHTML 1.1 se nehodí pro svou nepřístupnost pro staré prohlížeče, zbylé dvě varianty XHTML 1.0 jsou zase příliš volné, což, podobně jako u HTML klade velké nároky na disciplínu kodéra.

Co je XHTML

XHTML je jiná, svého času novější, norma jazyka HTML. HTML jako takové se dlouho nevyvíjelo, zůstalo ve verzi HTML 4.01, když přišel pokus zvaný XHTML.

To X na začátku XHTML znamená eXtensible, rozšiřitelný (ve skutečnost jde o zúžení a osekání).

Podstatné je, že podpora jazyka XHTML je v současných prohlížečích naprosto stejná jako podpora HTML (psáno 2004, platí i pro 2012). Ačkoli se usuzovalo, že v budoucnu bude podpora XHTML lepší než podpora HTML, na základě zkušeností s historickým vývojem prohlížečů není důvod se domnívat, že tomu tak vskutku bude.

Rozdíly XHTML oproti HTML

V XHTML na rozdíl od HTML musí být ukončené všechny tagy včetně nepárových jako jsou <meta>, <link>,
, <hr> nebo <img>. Zápis může mít více podob. Buď použijeme klasické (a validní) <img></img> nebo zkrácené <img/> nebo mírně upravené <img />. První způsob se nedoporučuje používat, zasíláme-li XHTML dokument s typem text/html. Druhý způsob - bez mezery - se nedoporučuje používat kvůli postarším prohlížečům,

které by v takovém případě mohly vynechat poslední atribut, je-li nějaký uvedený.

V XHTML na rozdíl od HTML musí být všechny tagy a jejich atributy zapsány malými písmeny, a to z toho důvodu, že jsou takto deklarované v odkazované DTD a X(HT)ML je case sensitive, tedy záleží na velikosti písem. Pokud bychom si deklarovali vlastní DTD, můžeme směle používat i velká písmena.

Všechny hodnoty atributů musí být uzavřeny do uvozovek.

Dokument musí začínat XML deklarací. Její použití není povinné, pokud je dokument kódován v UTF-8 nebo pokud určujeme kódování vyšší protokolem (http například).[14]

Pokud potřebujeme pracovat s rámy, můžeme deklarovat XHTML 1.0 Frameset a pro jednotlivé stránky XHTML 1.0 Transitional.

XHTML dokument bychom měli zasílat s jiným MIME typem než klasické HTML dokumenty.[15]

8.4. HTTP

HTTP je internetový protokol, původně určený k výměně hypertextových dokumentů mezi serverem a prohlížečem (tzv. služba WWW). Současná verze HTTP však již dokáže přenášet jakékoliv soubory a používá se i k mnoha jiným funkcím (např. spouštění vzdálených aplikací). K HTTP existuje také jeho zabezpečená varianta HTTPS.

HTTP funguje na principu **dotaz → odpověď**, jednotlivé dotazy nejsou z pohledu serveru rozeznatelné. Proto se HTTP také říká **bezstavový protokol**. To bylo výhodou v době jednoduchých internetových prezentací, při programování složitějších webových aplikací to však činí problémy, neboť HTTP např. neumožňuje uložení obsahu košíku v internetovém obchodě. To je pak nutné obcházet různými metodami, např. využitím cookies.

HTTP protokol

Http protokol je způsob, kterým si povídá prohlížeč se serverem, když se stahují stránky.

K čemu je dobré znát HTTP

Pro normální tvorbu stránek je znalost http protokolu vcelku k ničemu. Jakmile ale děláte složitější věci nebo se zajímáte o pokročilejší optimalizaci pro vyhledávače, je lepší vědět, co se mezi serverem a klientem vlastně děje.

Z http hlaviček můžete vyčíst např. informace o přesměrování, kešování, o cookies, o komprimaci nebo třeba o referreru.

Pokud píšete serverové skripty nebo složitější webové programy, je dobré vědět, kdy a jak posílat kterou http odpověď.

Zkratka HTTP znamená HyperText Transfer Protocol, tedy protokol přenášení hypertextu. (Hypertext je text s odkazy.)

Jak funguje http protokol

Povídá si klient se serverem. Klient něco chce a server mu to dá.

- Klientem je nejčastěji internetový prohlížeč (Explorer, Mozilla, Opera), ale může jím být třeba i vyhledávací robot nebo jiný program.
- Http server je program běžící někde v serverovně na nějakém počítači (ten počítač se náhodou též označuje jako "server", ale nikoli jako "http server"). Nejpoužívanějším http serverem je program zvaný Apache.

Takže http protokol je takový jakoby jazyk, kterým si povídají dva programy. Povídají si po síti, nejčastěji po Internetu.

Klient většinou chce nějakou stránku -- připojí se na server a požádá server o URL stránky. Tato žádost je formulována v HTTP protokolu (samotné připojení na server je přes TCP protokol). Server tuto http žádost zpracuje a pošle zpátky odpověď, která je psaná taktéž v HTTP protokolu. Například pošle klientovi http hlavičky a za nimi text stránky v HTML. Klient odpověď přijme, hlavičky si přečte a stránku zobrazí.

Příklad http komunikace

Čtenář si chce přečíst například tuto stránku. Tato stránka má URL

https://www.jakpsatweb.cz/server/http-protokol.html.

- Čtenář toto URL zadá do prohlížeče.
- Prohlížeč (tedy klient) si vyhodnotí doménu, přes DNS si zjistí, jaké IP adresy se má ptát.
- Přes TCP protokol naváže spojení se serverem na zjištěné IP adrese. Teprve nyní začíná HTTP.
- Prohlížeč pošle na server toto HTTP volání:

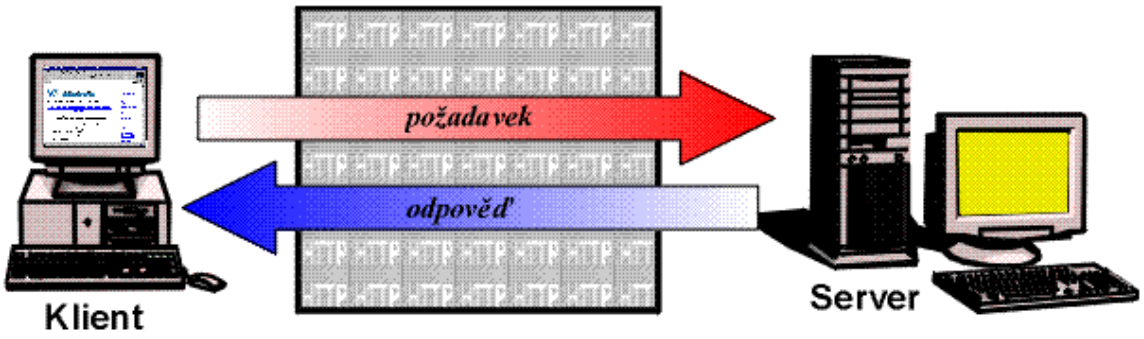
```
GET /server/http-protokol.html HTTP/1.1
HOST: www.jakpsatweb.cz
```

Základy protokolu HTTP

Služba World-Wide Web je postavena na třech základních technologiích -- HTML, URL a HTTP. HTML je značkovací jazyk, který slouží k standardnímu popisu obsahu a struktury webovských stránek. URL je speciální druh adres používaných na Webu -- každá webovská stránka má svoji jednoznačnou adresu právě v podobě URL. HTTP -- Hypertext Transfer Protocol -- je protokol používaný při komunikaci mezi prohlížeči a webovskými servery. Pomocí tohoto protokolu se k serveru přenáší URL stránky, kterou uživatel (přes svůj prohlížeč) požaduje, a naopak server pomocí protokolu HTTP odesílá uživateli zpět stránku zapsanou v HTML.

Pro správné pochopení principů CGI-skriptů je nezbytná alespoň základní znalost protokolu HTTP. Proto se v dnešním pokračování seriálu podíváme na základní vlastnosti protokolu HTTP.

Protokol HTTP vychází z architektury klient/server. Klient -- v našem případě prohlížeč -- se spojí se serverem a pošle mu požadavek. Server jako reakci na klientův požadavek zasílá odpověď. Přesný formát požadavku a odpovědi je definován ve specifikaci protokolu HTTP. Celou situaci mírně komplikuje to, že dnes existují tři verze protokolu -- 0.9, 1.0 a 1.1. Formát požadavku a odpovědi se v jednotlivých verzích odlišuje.



Obr. 1: Průběh komunikace mezi klientem a serverem

Základní rysy protokolu HTTP

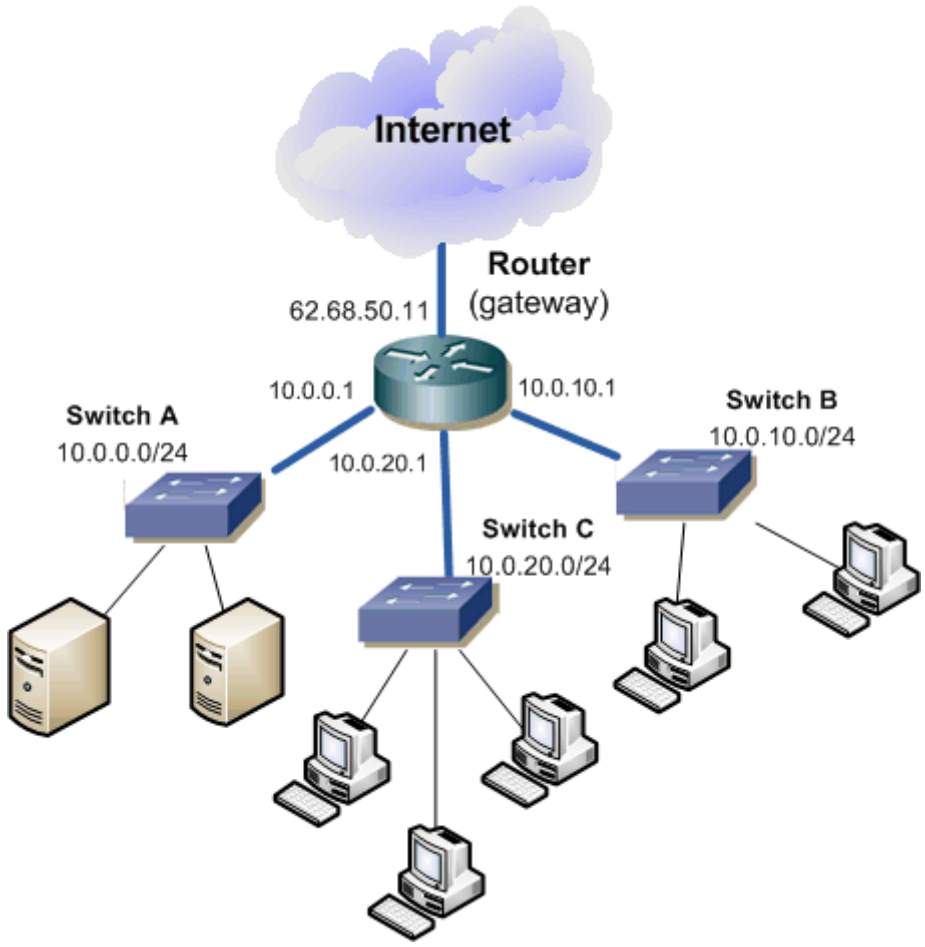
K úplnému pochopení článku budeme muset znát už trochu více základní rysy HTTP, a jak tento protokol vlastně pracuje. Protokol HTTP je protokolem aplikační úrovně pro distribuované hypermediální informační systémy. V praxi to znamená, že tento protokol se obecně na internetu používá nejen pro přenos dat mezi klientem a serverem, ale i pro mnoho dalších úloh. Protokol HTTP je bezstavový, tj. že nerozpoznává klienty, od nichž chodí požadavky. Pokud jeden klient odešle požadavek a vzápětí ten stejný klient odešle další požadavek, server nepozná, že jde o stejného klienta.

HTTP existuje ve 3 verzích a to 0.9, 1.0 a 1.1. První z nich, označována za HTTP/0.9 existovala jako jednoduchý protokol, který uměl v omezené podobě přenášet data na internetu. Verzí HTTP/1.0 nabyl protokol možnosti přenášení informací ve formátu MIME, takže mohl obsahovat i metainformace o přenášených datech. Nejpodstatnějším vylepšením protokolu verzí HTTP/1.1, což je zároveň poslední, aktuální verze, bylo to, že všechna spojení se stala trvalými. To znamená, že se spojení uzavře, až když jeden z dvojice klient-server odešle hlavičku pro uzavření. Dříve HTTP uzavíralo spojení po každé odpovědi serveru. Tímto vylepšením se také nesrovnatelně zvýšila rychlost přenosu, protože server už pro každý obrázek, rám a applet nemusí otevírat nové spojení.

9. Směrovací protokoly

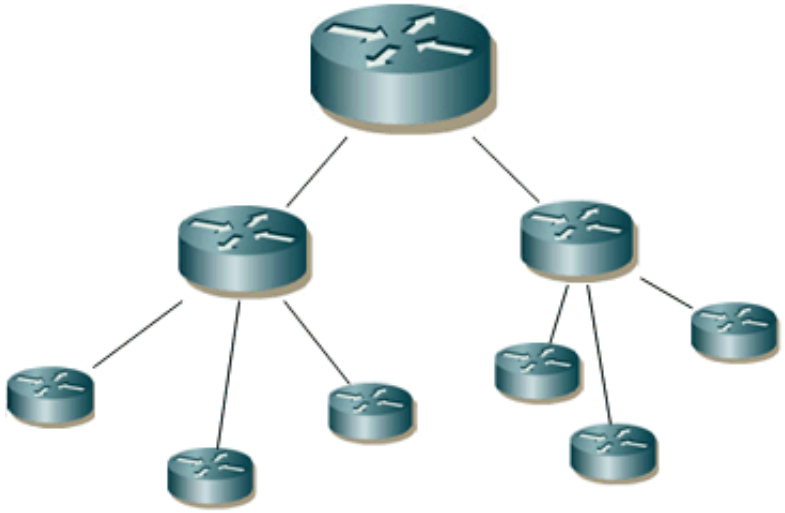
Routing, česky řečeno směrování, ale častěji se používá slovo routování (alespoň v mém okolí). Jedná se o techniku, která slouží k propojení jednotlivých sítí (přesněji subnetů). Původním zařízením, určeným pro routování byl router, ale v dnešní době se velmi využívají L3 switche, firewally nebo pouze servery/počítače. Router přeposílá komunikaci z jedné sítě do jiné.

Následující obrázek ukazuje jednoduchý příklad sítě, kde máme subnety A, B a C. Tyto subnety jsou propojeny přes router mezi sebou a také do internetu. Takže pokud chce například stanice ze subnetu B komunikovat se serverem v subnetu A, tak pošle data na router a ten zařídí doručení do subnetu A. Pokud by stanice chtěla komunikovat do internetu, tak router naopak zašle data na jiný interface.



Dělení sítě na subnety je hierarchické a ve všech propojích musí být router. Při komunikaci pak postupujeme směrem nahoru ve stromu na nejbližší vrstvu, která propojuje dané subnety, a pak opět dolů. Délka cesty se počítá podle počtu **hopů** (skok), což je každý přechod ze zařízení na zařízení, je to tedy počet routrů v cestě + 1. Přímé spojení dvou počítačů je dlouhé 1 hop. Používá se také termín ***next hop*** (další skok) a označuje se jím adresa dalšího routeru v cestě.

Na dalším obrázku jsem se pokusil zachytit tuto situaci. Je zde malý (a pouze schematický) výřez větší sítě (či internetu). Na listech stromu jsou malé routery, ke kterým jsou připojeny switche a počítače. Tyto routery se sdružují do dalších (větších) a tak dále na několika úrovních. Samozřejmě v praxi jsou větší routery vždy redundantně, aby byla síť odolná vůči výpadku či kvůli vyvažování zátěže.



9.1. Důležité pojmy pro routování

Router

Zařízení, které provádí routování.

Routing

Routování, přeposílání (forwarding) dat mezi sítěmi.

Route

Cesta, která se použije, zapsaná v routovací tabulce.

Routing table

Routovací tabulka obsahuje záznamy o jednotlivých cestách.

Routing protocol

Routovací protokol slouží ke směrování routovaného protokolu, určuje nejlepší cestu k cíli a posílá routovací informace dalším routrům.

Routed protocol

Routovaný protokol je IP, IPX nebo Apple Talk.

Ještě jeden občas používaný termín, který je dobré znát.

Router on stick

Je router, který je připojený do switche pomocí jednoho trunk portu - tzn. máme pouze jeden router a pouze jednu linku, což přináší velkou zátěž na router i linku a problémy při výpadku.

Dělení routovacích protokolů

Do routovací tabulky se vytváří několik typů záznamů cest (route), záleží na tom, jakým způsobem vznikly. Pakety jsou podle toho směrovány jedním ze základních způsobů routování:

- statické routování – ručně zadané cesty (záznamy v routovací tabulce), bezpečné a dobré, ale nereflektuje změny v topologii sítě,
- dynamické routování - síť se automaticky přizpůsobuje změnám v topologii a dopravě, automaticky se vypočítávají cesty pomocí routovacího protokolu,
- defaultní routování - pokud neexistuje jiná cesta, tak se použije defaultní.

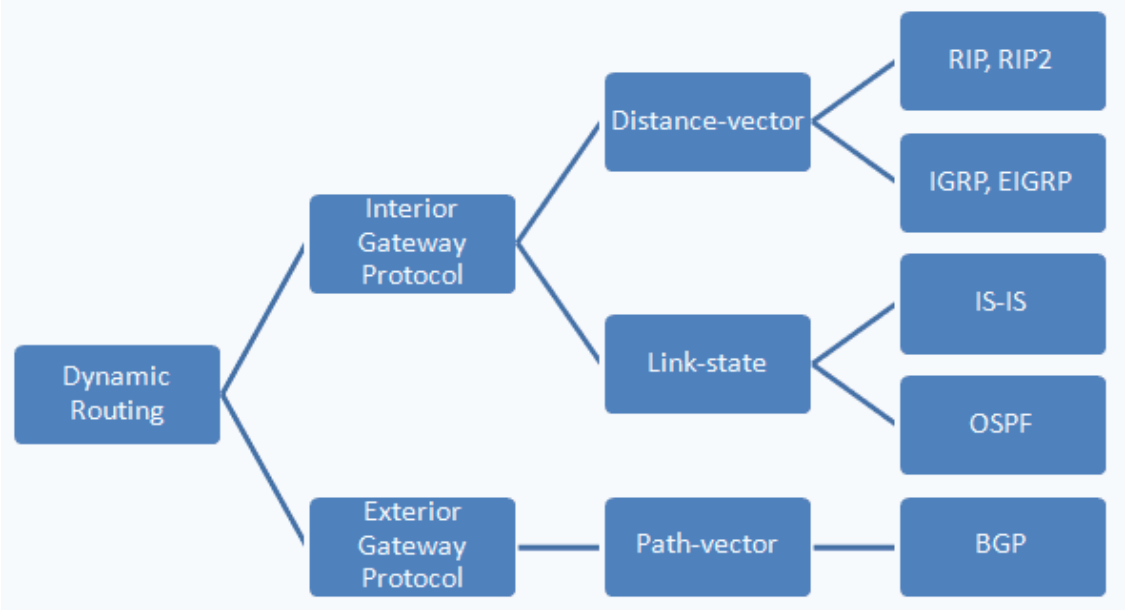
Dynamické routovací protokoly jsou dvou základních typů

- distance-vector routing protocol – routery udržují routovací tabulku s informací o (vektoru) vzdálenosti do dané sítě, periodicky routovací tabulku zasílají sousedům, ti si upraví svoji tabulku a tu opět odešlou dál, pro výpočet nejlepší cesty se používá jedna (počet hopů u RIP) nebo více metrik (propustnost linky a zpoždění u IGRP). Upraveným typem distance-vector protokolu je path-vector protocol.
- link-state routing protocol – routery udržují komplexní databázi síťové topologie (vytvořenou pomocí LSA), vyměňují si link-state advertisements (LSA), LSA jsou vyvolány nějakou událostí v síti, do svého okolí také odesílá Hello pakety, kde zasílá informace o sobě, rychle reaguje na změny topologie, ale spotřebovává více pásma a zdrojů na routeru, metrika je komplexní, nejlepší cesta se počítá pomocí Dijkstrova algoritmu shortest path first (SPF).

Pozn.: Ještě je zde jeden speciální typ, který vychází z distance-vector protokolu a přidává některé vlastnosti link-state protokolu, označuje se jako hybrid routing protokol nebo advanced distance-vektor protokol. Jeho jediným zástupcem je EIGRP.

Dále dělíme dynamické protokoly podle toho, zda jsou určeny pro nasazení uvnitř lokální sítě (přesněji řečeno uvnitř autonomního systému (AS), který může obsahovat několik LAN) nebo fungují napříč sítěmi (spojují AS dohromady)

- interior gateway protocol - IGP - routuje uvnitř Autonomous System (AS),



- exterior gateway protocol - EGP - routuje mezi AS.

10. BEZPEČNOST A ŠIFROVÁNÍ

10.1. Bezpečnost sítí

Nebezpečí:

Odposlech, modifikace přenášených dat, neoprávněný přístup do lokální sítě.

Je třeba chránit:

- data (zajistit, aby je nemohl někdo získat, měnit či mazat),
- výpočetní kapacity jednotlivých uzlů,
- omezování funkčnosti či narušování provozu některých služeb.

Pasivní útoky

- „odposlouchávání " dat - cílem získat nezveřejňované informace, které lze zneužít,
- monitorování provozu - analýzy takto provozovaných kontaktů.

Aktivní útoky

- modifikace dat,
- vytváření falešných dat,
- aktivním útokům nelze 100% zabránit, ale lze je na rozdíl od pasivních útoků snadněji detekovat.

Cíle bezpečnostních služeb

- zajištění důvěrnosti dat - pomocí šifrování celého komunikačního kanálu nebo jen vybraných citlivých dat,
- zajištění autentizace uživatelů sítě (odhalování maskovaného narušitele),
- zajištění integrity dat,
- zajištění neodmítnutelnosti zpráv - zajistit, aby odesílatel nemohl popřít odeslání zprávy a příjemce nemohl popřít přijetí zprávy,
- přiřazování přístupových práv - cílem je omezit (a řídit) přístup k počítači, datům a aplikacím, součástí je identifikace a autentizace toho, kdo žádá o přístup,
- zabezpečení dostupnosti síťových služeb - útokům na dostupnost služeb lze zabránit autentizací a šifrováním.

10.2. Firewall

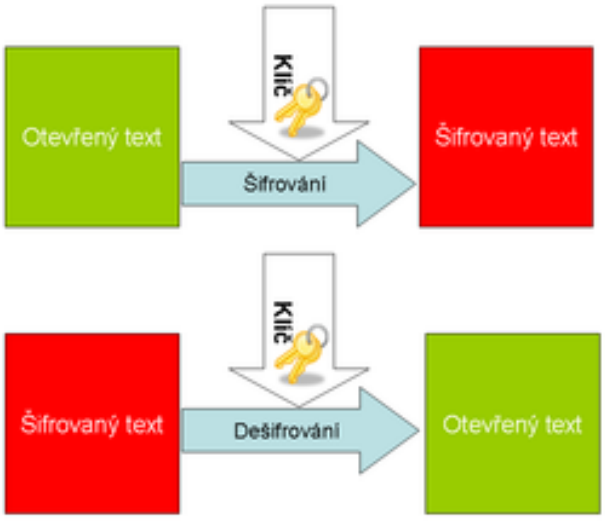
- je soubor opatření (realizovaný určitým HW a SW), která zabezpečují síť proti neoprávněnému přístupu zvenčí a proti úniku informací
- umožňuje např.: řízení přístupu uživatele z vnější i vnitřní sítě, nastavení přístupových práv, odfiltrování nebezpečných služeb, soustředění bezpečnosti do jednoho komunikačního uzlu, zablokování nepřátelského mapování vnitřní sítě, audit legálních a nelegálních operací
- zajišťuje bezpečnost při vstupu nebo výstupu do i ze sítě
- plní funkci filtru - rozhoduje o tom, co a kam bude přes něj propuštěno

10.3. Šifrování

1. Symetrická šifra

Šifrování a dešifrování pomocí jediného klíče. Symetrická šifra, někdy též nazývaná konvenční, je takový šifrovací algoritmus, který používá k šifrování i dešifrování jediný klíč. Tím se liší od algoritmů s veřejným klíčem, které mají dvojici klíčů – tajný a veřejný.

Podstatnou výhodou symetrických šifer je jejich nízká výpočetní náročnost. Algoritmy pro šifrování s veřejným klíčem můžou být i stotisíckrát pomalejší. Na druhou stranu velkou nevýhodou je nutnost sdílení tajného klíče, takže se odesilatel a příjemce tajné zprávy musí předem domluvit na tajném klíči.

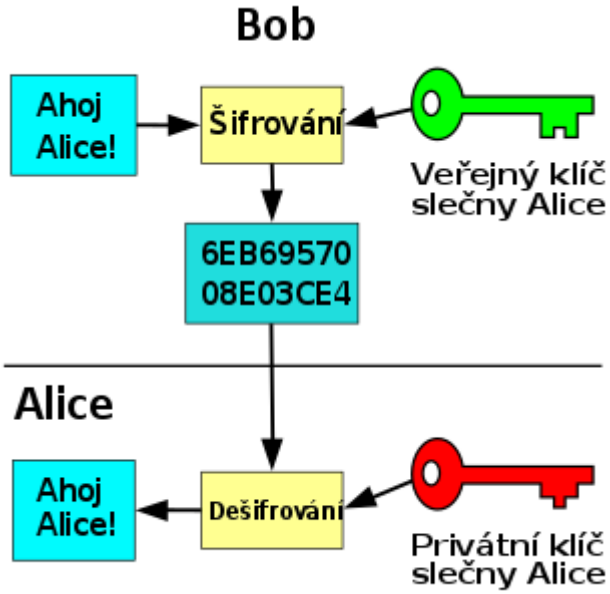


Obrázek 2: symetrické šifrování

2. Asymetrická kryptografie

Příklad asymetrického šifrování. **Asymetrická kryptografie** (**kryptografie s veřejným klíčem**) je skupina kryptografických metod, ve kterých se pro šifrování a dešifrování používají *odlišné* klíče. To je základní rozdíl oproti symetrické kryptografii, která používá k šifrování i dešifrování jediný klíč.

Kromě očividné možnosti pro utajení komunikace se asymetrická kryptografie používá také pro elektronický podpis, tzn. možnost u dat prokázat jejich autora.



Obrázek 3: asymetrické šifrování

Šifrovací klíč pro asymetrickou kryptografii sestává z dvou částí: jedna část se používá pro šifrování zpráv (a příjemce zprávy ani tuto část nemusí znát), druhá pro dešifrování (a odesilatel šifrovaných zpráv ji zpravidla nezná). Je vidět, že ten, kdo šifruje, nemusí s dešifrujícím příjemcem zprávy sdílet žádné tajemství, čímž eliminují potřebu výměny klíčů; tato vlastnost je základní výhodou asymetrické kryptografie.

Nejběžnější verzí asymetrické kryptografie je využívání tzv. veřejného a soukromého klíče: šifrovací klíč je veřejný, majitel klíče ho volně uveřejní, a kdokoli jím může šifrovat jemu určené zprávy; dešifrovací klíč je privátní (tj. soukromý), majitel jej drží v tajnosti a pomocí něj může tyto zprávy dešifrovat (existují i další metody asymetrické kryptografie, ve kterých je třeba i šifrovací klíč udržovat v tajnosti).

Je zřejmé, že šifrovací klíč **e** a dešifrovací klíč **d** spolu musí být matematicky svázány, avšak nezbytnou podmínkou pro užitečnost šifry je praktická nemožnost ze znalosti šifrovacího klíče spočítat dešifrovací.

Bezpečnost počítačových sítí

Sítí budeme rozumět soustavu několika výpočetních systémů, uživatelé přistupují k síti prostřednictvím některého z těchto systémů.

- Sdílení - potenciální přístup má velmi velké množství lidí, různé stroje mohou být řízeny různými ne nutně bezpečnými systémy.
- Složitost - v síti se vyskytují nejrůznější operační systémy komunikující spolu via spojovací mechanismus, který by měl zajišťovat ochranu, tento mechanismus však musí být dostatečně obecný, navíc síť jako celek nelze podrobit testování či dokonce certifikaci.
- Neznámý perimeter - nikdy nevíme, kdo všechno je připojen, není jasné, jak se ostatní stroje chovají.
- Množství zranitelných míst - je nutné uvěřit bezpečnostním mechanismům na všech strojích, mnohé části sítě leží mimo jakýkoliv dohled provozovatelů.
- Neznámá cesta - většinou nelze ovlivnit, kudy budou data přenášena, tedy není k dispozici žádná informace, kdo s nimi může přijít do styku Ochrana komunikace principiálně je možné chránit komunikaci jakožto:
    - proud dat – někdy nazýváno jako „stream enciphering“, tj. šifrování proudu dat, kdy se vytváří dojem, že komunikační kanál je spolehlivý z hlediska možného útoku
    - jednotlivé zprávy – odpovídá dnes moderní volné vazbě systémů pomocí „messagingu“, šifrují se aplikační zprávy, nebo jejich relevantní části proudové šifrování lze provádět mezi dvěma uzly sítě, nebo mezi dvěma aplikacemi běžícími na těchto uzlech Šifrování na úrovni linky (Link Encryption) data jsou šifrována těsně před vstupem do komunikačního media, dešifrována ihned po příchodu na druhý počítač toto šifrování probíhá na úrovni fyzické případně linkové vrstvy referenčního modelu výhodou je, že tento mechanismus je pro uživatele transparentní a může být i velmi rychlý, navíc je snadno připojitelný k stávajícím zařízením

End-to-End šifrování

poskytuje kryptografickou ochranu po celou dobu přenosu toto šifrování probíhá přibližně na úrovni aplikační nebo prezentační vrstvy referenčního modelu toto šifrování však již nebývá transparentní a má-li být účinné, musí být vhodně zakomponováno do celého systému další výhodou je, že není nutno šifrovat veškerou komunikaci, ale pouze citlivá data na rozdíl od šifrování linky je schopno zajistit autentizaci a integritu (end-to-end) někdy jsou používány obě zmíněné metody zároveň - šifrování linky za účelem běžné preventivní ochrany dat a End-to End šifrování k docílení skutečně kvalitní ochrany senzitivních dat se zavedením šifrování souvisí nutnost existence mechanismu distribuce a správy nezbytných šifrovacích klíčů, potřebných centrálních autorit pro zajištění provozu systému kryptografické ochrany, vhodných kryptografických zařízení zajišťujících základní funkce kryptografické ochrany Kontrola přístupu v případě sítí přistupují k obvyklým problémům kontroly přístupu ještě následující okruhy.

Odstupňovaná přístupová práva

Přístup k senzitivním datům může být omezen na pouze některé uzly. Pokud i autorizovaný uživatel žádá o přístup z jiného uzlu, mohou jeho přístupová práva být

výrazně omezena, nebo může být zcela odepřen přístup k datům. Tichý modem (Silent Modem) Po přijetí volání modem nezačne bezprostředně generovat nosnou, ale počká, až se druhá strana pokusí o negotiation. Tím je přístup do jisté míry omezen pouze na uživatele, kteří vědí, že jde o linku vedoucí k počítači, metoda omezuje možnost náhodného nalezení tohoto portu. Obdobou tichého modemu může být v případě IP protokolu služba, dostupná na daném stroji na jiném než obvyklém portu.

11. Peer-2-peer sítě

Během posledních 10 let v oblasti stahování souborů z internetu velmi vzrostla úloha tzv. výměnných sítí typu peer-to-peer, z nichž je v poslední době asi nejpopulárnější síť s protokolem Bittorrent, která má řádově desítky miliónů uživatelů a desítky klientů (µTorrent, Vuze/Azureus). Fenoménem posledních 5 let je pak i rostoucí obliba veřejných komerčních webových serverů pro sdílení souborů (filehosting), z nichž nejznámější je patrně Rapidshare.com, ale v širším smyslu se sem dají zařadit i systémy, které řeší spolupráci více lidí na jednom dokumentu či synchronizaci souborů (Google Docs resp. Humyo.cz).

Sítě peer-to-peer

Pojem sítě peer-to-peer (P2P, „rovný s rovným“) je velmi obecný. Znamená vlastně jakoukoliv síť, kde probíhá nějaká symetrická komunikace či interakce mezi počítači (každý z nich umí iniciovat či naopak na základě vnější iniciace/požadavku vykonat potřebné transakce a operace). Nelze tedy její uzly funkčně rozlišit na dva rozdílné klasické druhy - klienty a servery. V důsledku toho je potlačena klasická centrální úloha serverů a vzniká jeden hybridní, univerzální druh počítačového uzlu – tzv. servent (často se i zde ze setrvačnosti používá název klient). Decentralizace Internetu, která začala již kdysi dávno v době Usenetu (1979) a Fidonetu (1984, systémy BBS), je tedy pomocí sítí P2P dovedena opravdu do důsledků. Rovnocennost všech počítačů v síti však neznamená, že je tato síť ve všech uzlech homogenní co do kvantitativních parametrů jako jsou např. rychlost připojení, objem sdílených dat, lokální konfigurace, rychlost procesingu apod.

Označení P2P se však v posledních letech stalo téměř mediálním synonymem právě pro internetové výměnné sítě (tj. systémy pro sdílení souborů), i když na tomto principu může fungovat i mnoho jiných typů aplikací – třeba aplikace pro distribuované výpočty, šíření novinek a zpráv, hlasová komunikace a chat (IRC, Skype, Qnext, DKMessenger) či P2P vysílání internetových rádií a TV stanic. Mediální stereotypy také v podstatě ztotožnily sítě P2P (pro sdílení souborů) s pirátskou činností. Aniž bychom se snažili postihnout, do jaké míry tomu tak skutečně je, je třeba poznamenat, že např. síť Bittorrent je s výhodou využívána i k legálnímu šíření velmi velkých programů a souborů (linuxové distribuce) a existují též torrentové katalogy a trackery, které evidují a pomáhají šířit jen legální (např. filmový) materiál charakteru public domain (http://www.legittorrents.info/, http://beta.legaltorrents.com/, http://www.publicdomaintorrents.com/, http://www.jamendo.com/en/), tj. ten, který je zcela legálně zdarma.

Společným znakem výměnných sítí P2P je vlastně jen to, že v jejich rámci lze sdílet soubory či textové zprávy uložené na kterémkoliv počítači, připojeném k Internetu (a samozřejmě vybaveném běžícím klientem/servem dotyčné sítě). Jinak se sítě technicky značně liší, jak co do reálného stupně své funkční symetrie resp. homogenity a (de)centralizace, tak co do stupně relativní "anonymity". (Dokonalá anonymita je na Internetu samozřejmě nemožná.) Podle těchto znaků se někdy zhruba rozlišují jejich

různé generace.

11.1. Generace sítí P2P

První generací byly sítě, které měly uloženy seznamy a adresy souborů i počítačů na speciálním centrálním serveru či jejich větším počtu (Napster, OpenNap, chat – IRC (již od r. 1988)/IRC@find, Soulseek – ten poslední dodnes funguje a je používán relativně menší komunitou fanoušků hudby). Evolučně nejstarší výměnné P2P sítě tedy používají pro část svých činností centralizovanou strukturu typu klient-server. Zejména jde o připojení k síti a o vyhledávání zdrojů (souborů) na ní. Pro rutinní komunikaci a sdílení mezi koncovými klienty už serverů není třeba ani zde, to probíhá ryze formou peer-to-peer.

Sítě druhé generace jsou ty, které dnes nejčastěji potkáváme. Zde chybí centrální servery, ale ve skutečnosti nejsou tyto sítě drtivou většinou založeny jen na úplně rovnocenných a stejných serventech-uzlech, tj. každý počítač není v síti kvalitativně úplně nahraditelný kterýmkoliv jiným (výjimkou jsou jen některé, např. síť s protokolem Gnutella či Freenet). V praxi se z dokonalé symetrie v rámci principu peer-to-peer často slevuje, a to kvůli zefektivnění a větší spolehlivosti hledání či identifikace souborů a zrychlení jejich stahování.

Dnešní systémy P2P se proto často vyznačují "lokálně-centralistickými" či speciálně delegovanými/dedikovanými prvky, jako jsou různé "superuzly" (síť FastTrack s klienty jako Kazaa), huby (DirectConnect, DC++) nebo hledače a indexátory (OpenFT), kterými se mohou (volitelně) stát všechny jednotlivé počítače. Tyto speciální „naduzly“ jsou důležité pro rychlé napojení a katalogizaci všech ostatních, řadových počítačů-uzlů a jejich zdrojů a pro předcházení výskytu „úzkých hrdel“ v tocích dat. Dalšími "asymetrickými" prvky bývají specializované úložné servery, poskytující digitální hashovací otisky jednotlivých souborů (identifikační signatury, např. torrenty-BitTorrent, magnet linky – zejména Gnutella, linky ed2k – eDonkey/Overnet), resp. servery, pomáhající koordinovat zrychlené "rojové" stahování jednotlivých parciálních datových segmentů mezi počítači (trackery).

Přesto novější sítě používají strukturu typu peer-to-peer víceméně pro všechny účely a úkoly a proto se jim také říká "skutečné sítě P2P". Skutečné sítě typu peer-to-peer mají navíc tu příjemnou vlastnost, že jejich celková přenosová či komunikační kapacita pro průměrného uživatele s růstem počtu uživatelů/uzlů stoupá, nikoliv klesá, jako u centralizovaných systémů. To je navíc prohloubeno technologiemi pro segmentové stahování.

Druhá generace je také typická svými víceprotokolovými klienty, které umějí vyhledávat a stahovat soubory v rámci více sítí (protokolů), někdy i najednou. Jde např. o klienty MLDonkey, KCeasy a Shareaza. Speciální otázkou jsou tzv. překryvné protokoly, vytvářející jakousi „nadsíť“ v rámci dané sítě (Overnet).

Vynořující se třetí generace sítí a příslušných klientů zavádí či posiluje prvky šifrování

(maskování datového provozu, to je volitelně obsaženo už u sítě BitTorrent) a anonymity či pseudonymity, tj. utajení IP adres počítačů-uzlů (šifrování vzájemného routingu či linkování). Také se snaží o ještě větší decentralizaci, než je obvyklá dnes (Freenet, GNUnet). Na některé tyto sítě se nedostaneme bez schválení ostatními uživateli (sítě typu friend-to-friend - ANts P2P, WASTE, MUTE) Jiné se blíží svým charakterem téměř tzv. virtuálním privátním sítím (VPN), např. I2P, na nichž je možno provozovat veškeré internetové činnosti důvěrně. Někdy tyto sítě nezahrnují jen prosté sdílení souborů, ale kladou důraz i na chráněný prostor pro komunikaci a publikaci dokumentů bez možnosti cenzurních zásahů, čímž se stávají decentralizovaným, chráněným a privátním groupwarem. Záporem těchto systémů bývá menší uživatelský komfort a relativní pomalost.

Zmíněná internetová rádia a TV vysílání (souborně internetové streamy) typu peer-to-peer se někdy řadí do P2P aplikací čtvrté generace (TVUPlayer, PPLive, PeerCast, PPStream). Jistá část systémů P2P-TV je založena na bázi protokolu BitTorent nebo na protokolech podobných (zde však často jde o televizi "on-demand", ne o streaming v reálném čase), avšak ty hlavní používají systémy vlastní, odlišné. Celkově se takto daří vysílat desítky až stovky TV kanálů (často sportovních) a to s poměrně značnou účinností (danou tím, že každý přijímající uzel může být současně i retranslačním vysílačem pro uzly další).

11.2. Filehosting

Hostování souborů na (veřejných a komerčních) webových serverech je naproti tomu službou spíše technologicky „zastaralou“ a konzervativní, přesto při dnešních rychlostech připojení velmi účinnou. V této oblasti existuje nabídka snad přes 100 důležitých serverů zahraničních a možná až 10 serverů českých. Podmínky a systémy jejich užívání na straně downloadu i uploadu se individuálně různí, mají však často dva různé downloadovací módy – kapacitně či časově omezené stahování zdarma (placeno reklamou a omezeno přes OCR/captcha systémy, dočasné linky a časové intervaly) a „prémiové“, daleko méně omezené či prakticky neomezené stahování za jistý uživatelský poplatek. Uveďme si zde jistý zkrácený výběr těchto serverů - bez doménových přípon, které si jistě snadno doplníte v případě zájmu sami.

Zahraniční:

Rapidshare, depositfiles, filefactory, megaupload, mediafire, sendspace, uploading, zshare, ifolder, hotfile, icefile, letitbit, filefront, ifile, easy-share.

České:

Edisk, uloz.to, czshare, leteckaposta, quickshare, bagruj, nahraj.

V této oblasti se vyskytlo několik programů, které stahování z těchto serverů mohou do značné míry automatizovat a tedy ulehčit – např. Universal Share Downloader (USD) nebo RapGet. Další zajímavé odkazy:

- www.filesharing.eu Souhrn o sdílení souborů přes sítě P2P
- www.slyck.com Novinky a portál Slyck
- www.zeropaid.com Novinky a portál Zeropaid
- www.filesharingz.com Novinky
- www.p2pnet.net Novinky
- www.p2pforums.com Diskusní a novinkový portál
- www.infoanarchy.org "Decentralizační" P2P wiki-portál
- www.openp2p.com O´Reillyho stránka , věnovaná oblasti P2P
- www.planetpeer.de Portál pro anonymní sítě nejen pro sdílení souborů (MUTE, I2P, Freenet apod.)
- www.fileshareworld.com Rozcestník k výměnným systémům a P2P klientům
- www.ftc.gov/bcp/workshops/filesharing Názory americké Federal Trade Commission na sdílení souborů
- http://cs.wikipedia.org/wiki/Peer-to-peer České heslo na Wikipedii
- http://p2ptv.yourglobaltv.com/ a http://www.tvavailable.com/P2P_TV/ Přehledy nejpoužívanějších klientů P2P TV
- http://en.wikipedia.org/wiki/P2PTV Základní článek o P2P TV
- http://www.yourglobaltv.com/p2pchannels/ Hlavní sportovní kanály P2P televize
- http://en.wikipedia.org/wiki/File_hosting_service Hostingové servery obecně
- http://www.dimonius.ru/dusd.php - Universal Share Downloader (USD) - automatický stahovač
- http://www.rapget.com/en/ RapGet – něco jako předešlý USD.

12. Anonymita na internetu

Anonymita v prostředí internetu je v poslední době velmi diskutované téma dokonce i na tak vysokých úrovních, jako jsou vládní subjekty vyspělých států. Anonymita byla vždy důležitým faktorem v historii a to nejen moderní. Nicméně s příchodem nových technologií a jejich penetrací do společnosti je toto téma ožehavější než kdy dříve. Jednotlivé státy na jednu stranu podnikají kroky k získání kontroly nad internetem se snahou identifikovat uživatele kvůli bezpečnosti (obavou je především terorismus, dětská pornografie, prodej drog a praní špinavých peněz). Však na druhou protichůdnou stranu se státy snaží zachovat soukromí uživatelů na internetu a ochránit jejich citlivé osobní údaje, což dokládá například i prosazení tzv. „sušenkového zákona“ evropskou unií.

Pro dosažení dokonalé sociální anonymity je potřeba neznalosti ani jedné ze sedmi dimenzí identifikačních informací:

- Jméno osoby
- Lokace
- Pseudonym spojitelný s reálným jménem nebo lokací
- Pseudonym prozrazující jiné informace
- Odhalující vzorce chování
- Členství v některé sociální skupině
- Informace předmět či dovednost naznačující osobní charakteristiky

Důvody pro využívání anonymity zůstávají stejné jak v reálném prostředí, tak i v tom virtuálním. Jedná se o vyhnutí se důsledkům svého jednání (strach z represe, nepochopení atd.).

Identifikační technologie

Moderní technologie nabízejí mnoho možností a způsobů, jak jedince identifikovat.

IP adresa

Jedná se víceméně o unikátní adresu každého zařízení, které je připojeno do sítě. Z IP adresy je možné zjistit například informace o geografické poloze.

Geolokace

V poslední době jsou na vzestupu techniky, které dokáží zjistit polohu některé i s přesností až jeden metr. Například:

- Constraint-based geolocation – pracuje na základě aktivního měření vzdálenosti za pomocí odezvy.
- GPS – především v mobilních zařízeních
- Obsahová analýza příspěvků na webu (sociálních sítích) - jedná se o metodu, která podle analýzy veřejně dostupných příspěvků určí lokaci uživatele například podle zmíněných měst, státu ale i počasí v obsahu příspěvků. Také se bere v potaz i aktivita uživatele z pohledu času pro určení jeho časové zóny.

Cookies

Cookies slouží jako trvanlivý identifikátor mezi klientem a serverem. Z důvodu jejich citlivé povahy vůči soukromí se jimi zabývá Evropská unie.

PRISM

jedná se o vládní projekt americké tajné agentury NSA. Tento projekt má neomezený přístup k datům společností Google, Microsoft, Yahoo a spousty dalších, s tím, že 98% dat tvoří data právě trojice výše zmíněných společností. Společnosti však tento neomezený přístup k datům popírají.

12.1. Soukromí podporující techologie

Tor jedná se o projekt, který funguje na takzvaném konceptu Onion Routing a zajišťuje tak anonymizaci uživatele při pohybu na internetu. Program je určen k ochraně osobních údajů uživatelů, jejich svobody, soukromí a možnosti provádět důvěrné obchodování tím, že je chrání před sledováním jejich aktivit na internetu. Nicméně tento software může být využíván i k nelegálním činnostem, jelikož při použití této technologie je pak velmi obtížné pachatele dopadnout.

Orbot jedná se o verzi Toru pro mobilní telefony s operačním systémem Android.

Freenet Project tato technologie je vnímána jako velice bezpečná, anonymní a decentralizovaná platforma pro sdílení dat zaměřená proti cenzuře. Každý uživatel může vyhradit určitý prostor na svém lokálním disku, který je následně k dispozici celé síti k ukládání zašifrovaných souborů od ostatních uživatelů. Není tak nutné provozovat svůj vlastní server.

Internet je obrovskou sítí komunikačních kanálů, kterými proudí naše data. Každý balíček těchto dat, tzv. **paket**, obsahuje kromě samotného obsahu, který chceme někam poslat, také určité poznávací znaky, které zajímají „*Velké bratry*“. Těmi jsou mimo jiné **Google, Microsoft, Facebook, Twitter** a mnoho dalších.

Využívají **poznávací znaky** našich dat, aby si udělali obrázek o tom, na které webové stránky chodíme, co nás na nich zajímá, jak často je navštěvujeme a také, kde se při tom nacházíme. Všechny tyto informace následně vyhodnocují za **účelem cílené reklamy**. S nástupem sociálních sítí se **špehování** dostalo ještě dál.

Velcí bratři si mohou naši aktivitu na webu **kombinovat** s aktivitou našich známých a vytvořit si tak komplexní mapu uživatelů internetu. Tato **dobrovolná ztráta soukromí** je cenou za poskytování služeb „*zdarma*“. Abychom však zmíněné poskytovatele služeb jenom nešpinili, je nutno uznat, že jejich služby jsou po technologické stránce **velmi náročné a propracované**, i když nám to na první pohled nemusí být úplně zřejmé.

12.2. Základní principy zachování anonymity

Jak bylo řečeno, na internetu jsou sledovány především údaje o tom, **odkud** naše data proudí a že **patří nám**. Základní snahou by tedy mělo být **maskování těchto údajů**. K tomu mohou posloužit VPN, neboli v překladu Virtuální soukromé sítě, Proxy servery nebo speciální internetové prohlížeče. Jaké výhody a úskalí jednotlivá řešení přinášejí, se dočtete v následujících odstavcích.

Připojení přes VPN

S virtuálními soukromými sítěmi se nejčastěji setkáváme v zaměstnání, kde umožňují přístup více počítačů k sítovým úložištím, tiskárnám nebo jiným serverům. Zároveň však zprostředkovávají komunikaci s veřejným internetem. Pro webové stránky a velké bratry jsou pak **naše pakety označeny IP adresou** serveru VPN a nikoliv našeho počítače.

Pokud se tedy přihlásíme do sítě VPN, která je v jiném státě, nebude zjistitelná naše fyzická poloha. Navíc VPN server komunikaci šifruje, takže si nikdo, kromě koncového serveru nebude moci vaše data přečíst.

Na internetu existuje **mnoho VPN z různých koutů světa**, které jsou určeny pro anonymní surfování na internetu. Bohužel všechny **musejí zveřejnit vaší IP adresu**, pakliže jim to soud přikáže, jinak by byly ukončeny. Nelze proto spoléhat na to, že i když si předplatíte VPN, nikdo už nebude moci vystopovat vaši IP adresu.

Připojení přes proxy server

Podobně jako VPN i proxy server je p**rostředníkem ke komunikaci mezi naším PC a internetem**, a proto se ve světě internetu naše data budou tvářit, že pochází od něj. Na rozdíl od VPN však data **nebudou zašifrovaná**. Jelikož se jedná o jednodušší řešení než VPN, lze využít i proxy servery s veřejně přístupným webovým rozhraním.

Ty jsou výhodné v případě, kdy je **chceme využít jak krátkodobě**. Stačí do jejich adresního řádku vložit adresu, na kterou se chceme dostat. Proxy server jednoduše zprostředkuje naši komunikaci s daným webem.

Je třeba mít ale na paměti, že přes proxy server prochází všechna naše data a v případě nedůvěryhodného proxy **mohou být zneužita**. Nikdy proto nevyužívejte webové proxy k navštěvování zabezpečených webů (https).

Speciální webový prohlížeč a vyhledávače

Smyslem anonymních webových prohlížečů je to, co by **laik očekával od svého běžného prohlížeče**, když jej přepne do anonymního režimu. Ten totiž v žádném případě nemaskuje naši aktivitu na internetu před velkými bratry. Pouze před dalšími uživateli stejného počítače. Veškerá komunikace je stále stejně sledována, jako vždy.

Anonymní webové prohlížeče jsou naopak naprogramovány tak, aby **zabránily velkým bratrům ve sledování našich dat**. Toto řešení nepatří k neprůstřelným, ale zato je velmi **jednoduché a pohodlné**. Je výhodné kombinovat takový prohlížeč s anonymním internetovým vyhledávačem namísto známého Googlu, Yahoo! nebo Bing. Tyto vyhledávače totiž neshromažďují data o tom, co vyhledáváte.

Cibulové šifrování – Tor

Tor (The Onion Routing) je webový prohlížeč a také sít serverů. Často je Tor označován jako **internet v internetu**. Funguje na principu **cibulového šifrování**, kdy je každý paket šifrován ve vrstvách jako cibule a každý server umí rozšifrovat jen jednu vrstvu.

Každý paket tak projde přes určité množství (nejméně tři) servery než se dostane ke svému cíli. Každý Tor server přitom zná jen adresu příchozího a odchozího serveru a nikoliv celou kaskádu.

Proto je **téměř nemožné vystopovat paket až k jeho domovské IP adrese**. Tor si také vybudoval nelichotivou pověst, neboť slouží jako základ pro přístup do **černého internetu**, který je spojován s nelegálními aktivitami. Nevýhodou používání Tor prohlížeče je jeho nízká rychlost, neboť data cestují doslova sem a tam po celém světě.

12.3. TOR - totálně anonymní internet

Digitální stopa

Každý uživatel celosvětové sítě po sobě zanechává otisk své činnosti, kterému se již od osmdesátých let říká „digitální stopa“. O tom, jak velkou stopu po sobě na internetu zanecháme, rozhodujeme v první fázi do jisté míry sami - nastavením prohlížeče či instalací proxy serveru, jenž zprostředkovává komunikaci mezi vnitřní sítí a internetem a umožňuje řadu bezpečnostních nastavení. To jistě nezaručí úplnou anonymitu, ale pro běžné užívání internetu je to dostačující ochrana soukromí.

Větší ochrana

Někdy na začátku nového století se anonymita internetu dostala do popředí. Vzniklo několik projektů, které měly zajistit uživatelům co největší soukromí. Mezi nimi nejvíc vyčníval projekt sítě Tor (The Onion Routing) a to hlavně z důvodu, že ho sponzorovala a plně podporovala americká vládní agentura Naval Research Laboratory. Primárně měl fungovat jako ochrana vládní komunikace ve Spojených státech. Tor získal finanční podporu rovněž od neziskové organizace Electronic Frontier Foundation, která se zabývá ochranou práv v digitálním světě.

Uzel vedle uzlu

Jak to vlastně ve zcela anonymní síti funguje? Princip zabezpečení je založen na velkém počtu uzlů, přes které data putují. Veškerá komunikace je asymetricky šifrována a servery mezi sebou sdílejí veřejné klíče a vyměňují si generované dynamické klíče pro přenášené řetězce. Každý uzel je zcela autonomní a o tom, kam má řetězec vyslat dál, se dozví pouze z hlavičky zprávy. Tedy jeho informovanost o cestě datového toku končí u dalšího uzlu. Zjednodušeně to znamená, že server ví, komu co poslat a komu vrátit odpověď, ale tím jeho možnosti končí. Čím víc uzlů, tím větší anonymita. Uživatel sítě si sám určí jakou trasou a přes jaké uzly budou data putovat.

Falešné stopy

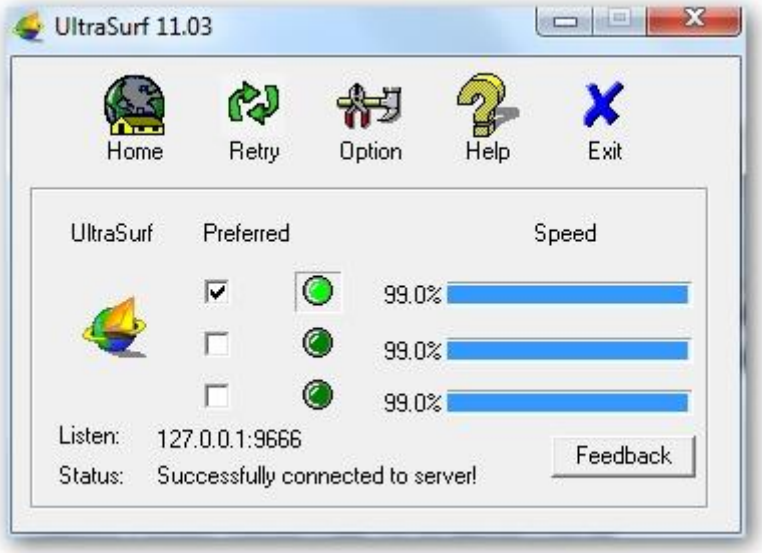
Komplikovanost přenosu dat přes anonymní servery by na první pohled mohla být dostačující, ale Tor jde dále. Servery mohou odesílat pakety v jiném pořadí, než je přijaly,

nebo mezi odesílané řetězce vložit falešné pakety, které mohou případné „sledování“ datového toku odklonit zcela mimo skutečnou trasu. Nevýhodou je delší odezva sítě Tor.

Jsou situace, kdy je potřeba, aby uživatel v zájmu zachování vlastního soukromí či dostupnosti některých služeb změnil svou IP adresu. Není to vůbec složitá operace. Řada služeb nabízí řešení prostřednictvím VPN a vlastního klienta s nadstandardními funkcemi, přístup k vlastním serverům, z nichž bývá komunikace přesměrována tak, že uživatel brouzdá po internetu v podstatě anonymně, nebo s možností šifrování.

UltraSurf

UltraSurf je minimalistický program, který umožní anonymní pohyb po internetu i opravdovým začátečníkům. Toho docílí především velmi jednoduchým nastavením, ovládáním, na systémové prostředky nenáročným provozem a dostupností zdarma. Program v malém okně nabídne připojení ke třem různým serverům.



Úspěšná změna IP adresy je oznámena ikonou v podobě zámku v nástrojové liště, dále lze v nastavení zapnout a vypnout ovládání pomocí klávesové zkratky, start prohlížeče, automatické vymazání cookies a historie a také nastavit proxy server. Pro přístup k americkým serverům a službám, které vyžadují tamní IP adresu, používá UltraSurf čtenář *freekarol*, podle nějž patří k těm nejlepším mezi bezplatnými aplikacemi.

proXPN

Tento VPN klient je velice populární, rozšířený a oblíbený, protože nabízí dobrý výkon, kvalitní službu, výběr mezi americkou, britskou, holandskou a singapurskou IP adresou, šifrování, nelimitovaný přenos dat, limitované ukládání informací o připojení (dva týdny) a spoustu dalších věcí. Program je možné využívat se zmíněnými službami zdarma, nebo si lze připlatit za bonus funkcí.

TunnelBear VPN

Služba a VPN aplikace TunnelBear bývá často svými uživateli označována za krásnou, velice jednoduchou a snadno ovladatelnou. Je navíc možné využívat jak bezplatnou verzi, kdy můžete přes „tunel“ protáhnout 500 MB měsíčně zdarma a při propagaci na Twitteru získáte další 1 GB dat navíc, tak verzi placenou a datově neomezenou za pět dolarů měsíčně.

Placená verze také umožní práci s touto VPN nejen na PC a Macu, ale také na mobilních zařízeních s iOS a Androidem.

CyberGhost VPN

Další doporučovanou VPN je CyberGhost, k němuž se šifrovaně připojíte (1024 bitovým SSL 128 bitovým AES heslem) pomocí softwarového klienta na virtuální privátní síť VPN a přes nějž pak prochází další komunikace do internetu. Program se snadno ovládá a během pár kliknutí dochází k připojení.

Síť serverů CyberGhost je dostatečně hustá i výkonná, také ztráta rychlosti je oproti jiným řešením výrazně menší. Program a službu lze využívat i zcela zdarma, ovšem s funkčními omezeními (1 GB přenesených dat za měsíc, rychlost do 2 Mb/s atd.). Autoři navíc nabízejí různě placené a různě výkonnostně odstupňované verze. CyberGhost doporučil *Suslikus*.

Hide My Ass

Celosvětově známá VPN, která disponuje širokou uživatelskou základnou i nabídkou služeb a serverů. Hide My Ass nabízí bez mála 40 000 IP adres v 53 různých zemích, za které se můžete schovat. Takovémuto počtu bezplatná řešení příliš konkurovat nemohou, takže za měsíc užívání je nutné zaplatit přes 11 dolarů, ovšem při dlouhodobějším předplacení lze cenu snížit až na šest a půl dolaru.


HMA! Pro VPN 2.7.0
HMA! Pro VPN
Secure internet encryption
Knowledge base
Email support
Live chat
Forums
NOT CONNECTED!
Your internet traffic is uncrypted
and your online identity is exposed
Dashboard
Country selection
IP address settings
Secure IP bind
Speed guide
Proxy settings
Billing & packages
Dashboard
Dashboard
Connection log (large view)
Settings
Login
Username
Password
Connect to VPN
Remember my username and password
Auto-connect after launch (requires a saved user/pass)
Launch on operating system startup
Start minimized
Protocol
OpenVPN
Quick VPN country change
USA, Texas, Dallas (LOC1 S1)
Full view
Connection Log
29/05/2011 15:38:10 LoadGroups: requesting (proxy type: none)...
29/05/2011 15:38:12 LoadGroups: success
29/05/2011 15:38:12 LoadServers: requesting (proxy type: none)...
29/05/2011 15:38:12 LoadServers: success
29/05/2011 16:08:14 Disconnect command issued: MANUAL, state: Disconnected
Diagnostics
Open log file

13. Útok a obrana na internetu

13.1. Útok a Obrana PC

13.1.1. ÚTOK

Viry

Název je odvozen díky jistým podrobnostem od biologických originálů. Virus je schopen seberegulace, tedy množení sebe sama, ovšem za přítomnosti vykonatelného hostitele, k němuž je připojen. Hostitelem mohou být například spustitelné soubory, systémové oblasti disku, popřípadě soubory, které nelze vykonat přímo, ale za použití specifických aplikací (dokumenty Microsoft Wordu, Skripty Visual Basicu apod.). Jakmile je tento hostitel spuštěn (vykonán), provede se rovně kód viru. Během toho okamžiku se obvykle virus pokouší zajistit další sebe-replikaci a to připojením k dalším vhodným vykonatelným hostitelům.

Trojské koně

Na rozdíl od virů není tento typ škodlivého kódu schopen sebe-replikace a infekce souborů. Trojský kůň nejčastěji vystupuje pod spustitelným souborem typu EXE, který neobsahuje nic jiného (užitečného), ne samotné tělo trojského koně. Odtud společně se skutečností, že trojan není připojen k žádnému hostiteli, plyne, že jedinou formou dezinfekce je odmazání dotyčného souboru. Starší definice říkají, že trojan je program vizuálně vypadající jako užitečný, ve skutečnosti však škodlivý. V daleké minulosti se tak několikrát objevil trojský kůň vydávající se za antivirový program McAfee virusScan, ve skutečnosti likvidující soubory na pevném disku. V současnosti se tak můžeme setkat nejčastěji s následující formou trojanů:

- Password-staling trojani (PWS)
- Destruktivní trojani
- Backdoor
- Proxy Trojan

Červ

Pojmem červ (worm) byl prvně označen tzv. Morrisův červ, který v roce 1989 dokázal zahltit značnou část tehdejší sítě, ze které později vznikl Internet. Tento a další červi (z poslední doby třeba populární Code Red, SQL Slammer, Lovsan / Blaster, Sasser) pracují na nižší síťové úrovni než-li klasické viry. Nešíří se ve formě infikovaných souborů, ale síťových paketů. Jmenované pakety jsou směrovány od úspěšně infikovaného systému na další systémy v síti Internet (a buď náhodně, nebo dle určitého klíče). Pokud takový paket dorazí k systému se specifickou bezpečností dírou, může dojít k jeho infekci a následně i k produkci dalších červích paketů. Šíření červa je tedy postaveno na

zneužívání konkrétních bezpečnostních děr operačního systému, úspěšnost pak od rozšířenosti daného softwaru obsahující zneužitelnou bezpečnostní díru. Z výše uvedených charakteristik plyne, že červy nelze detekovat klasickou formou antivirového softwaru. Vedlejším efektem může být kompletní zahlcení sítě, podnikové LAN nevyjímaje. Pojem červ je často spojován i s typem infiltrace šířící se elektronickou poštou. Zde tak mohou pojmy virus a červ splývat v jeden.

Spyware

Spyware je program, který využívá Internetu k odesílání dat z počítače bez vědomí jeho uživatele. Na rozdíl od backdooru jsou odcizovány pouze "statistická" data jako přehled navštívených stránek či nainstalovaných programů. Tato činnost bývá odůvodňována snahou zjistit potřeby nebo zájmy uživatele a tyto informace využít pro cílenou reklamu. Nikdo však nedokáže zaručit, že informace nebo tato technologie nemůže být zneužita. Proto je spousta uživatelů rozhořčena samotnou existencí a legálností spyware. Důležitým poznatkem je, že spyware se šíří společně s řadou sharewarových programů a jejich autoři o této skutečnosti vědí.

Adware

Obvykle jde o produkt, který znepříjemňuje práci s PC reklamou. Typickým příznakem jsou "vyskakující" pop-up reklamní okna během surfování, společně s vnucováním stránek (např. výchozí stránka Internet Exploreru), o které nemá uživatel zájem. Část Adware je doprovázen tzv. "EULA" - End User License Agreement licenčním ujednáním. Uživatel tak v řadě případů musí souhlasit s instalací. Adware může být součástí některých produktů (např. BSPlayer). Ačkoliv nás reklama doprovází během celé činnosti s daným programem, odměnou je větší množství funkcí, které nejsou v klasické free verzi (bez reklamy) dostupné.

Dialer

Dialer je program, který změní způsob přístupu na Internet prostřednictvím modemu. Místo běžného telefonního čísla pro Internetové připojení přesměruje vytáčení na čísla se zvláštní tarifikací, např. 60 Kč / minutu (tzv. "žluté linky"). Éra dialerů se ale týká pouze analogových tel.linek (dial-up) a netýká se ADSL a jiných moderních technologií.

Phishing

Tímto slovem se označují podvodné e-maily, kdy jsou na velké množství adres rozeslány podvodné dopisy, které na první pohled vypadají jako informace z významné instituce (nejčastěji banky). Tyto dopisy plně využívají tzv. sociální inženýrství. Příjemce je informován o údajné nutnosti vyplnit údaje v připraveném formuláři, jinak mu může být zablokován účet (v případě banky), popřípadě může být jinak znevýhodněn. V e-mailu bývá uveden odkaz na připravené stránky s formulářem, které jakoby odkazovaly na server této významné instituce. Ve skutečnosti je uživatel přesměrován na cizí server, ale vytvořený ve stejném designu, jako jsou stránky pravé instituce. Chycený uživatel nemusí poznat rozdíl a může vyplnit předvolená políčka, kde jsou po něm požadovány důvěrné informace, čísla účtu, kódy k internetovému bankovnictví, pin pro platbu atd. Takto získané údaje mohou podvodníci velice snadno zneužít.

13.1.2. OBRANA

Prevence je v dnešní době bohužel pojmem, kterému se příliš uživatelů PC nevěnuje, popř. věnuje jen částečně. Je důležité si uvědomit, že pouze nainstalovaný antivirový program (v lepším případě i pravidelně aktualizovaný a správně nastavený) je sám o sobě nedostatečnou prevencí. Obrovsky důležité jsou PRAVIDELNÉ AKTUALIZACE alespoň těch produktů, které mají něco společného s počítačovou sítí nebo Internetem. Ve stručnosti tak lze prohlásit, že je potřeba:

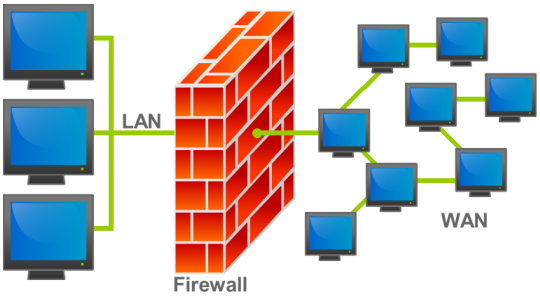
- povolit pravidelnou aktualizaci operačního systému Windows a jeho součástí (nabídka Start / Ovládací panely / Automatické aktualizace),
- pravidelně aktualizovat i další, běžně používané produkty, které by mohly být z Internetu zneužity (Mozilla Firefox, ICQ, DC++), po případě v nich přímo povolit automatickou aktualizaci. Toto se týka pochopitelně i různých "pluginů" pro prohlížeče (java, shockwave, flash player).

Antivirový systém

Antivirový systém by měl být nedílnou součástí každého PC. V drtivé většině případů je součástí antiviru i antispyware. U antiviru & antispyware je nutné zajistit pravidelnou - AUTOMATICKOU aktualizaci. Vzhledem k rychlosti šíření havěti (především prostřednictvím elektronické pošty), není aktualizace 2x za hodinu žádným luxusem. Obecně platí, že čím častěji se bude antivirový systém pokoušet o stažení aktualizace, tím lépe. Stejně tak je třeba zajistit minimálně chod modulu, starající se o nepřetržitou kontrolu souborů, se kterými uživatel nebo aplikace manipuluje (které otevírá/ ukládá/ spouští apod.). Takové moduly se většinou označují pojmy jako "rezidentní štít" či odborně "on-access skener". Je nutné si uvědomit, že tento modul je nejdůležitější součástí každého antivirového systému. Ve většině případů je sice nabízen i modul, který kontroluje přímo stahovanou či odesílanou poštu, ale i tento lze v nouzi oželet, jelikož tak jako tak by případné infekci zabránil on-access skener (až na některé výjimky způsobené zneužitím chyby v programu). Sice by virus neodchytil předtím, než bude zařazen do schránky doručené pošty, ale odchytil by ho v momentě, kdy by se uživatel pokusil otevřít - spustit infikovanou přílohu e-mailu.

Firewall

Firewall, nejčastěji v podobě tzv. personálního firewallu je vhodným doplňkem uživatele přistupujícího k Internetu. Přítomnost firewallu je téměř nutností v momentě, kdy je uživatel k Internetu připojen veřejnou IP adresou a jeho počítač je tak přímo dosažitelný odkudkoliv z Internetu (veřejnou IP obvykle dostane uživatel při dial-up připojení či při připojení přes kabelový Internet - typicky Chello/UPC). Podobně jako v případě antivirových systémů je ale potřeba, aby dokázal uživatel firewall správně používat. Musí být schopen určit, jaká komunikace je legální a ilegální, resp.: jakou povolit a zakázat. V opačném případě ztrácí přítomnost firewallu smysl a nezáleží přitom, zda používáte ten, integrovaný ve Windows, nebo jiný placený firewall třetí strany.



13.2. Nebezpečná síť - jak se útokům bránit

Na síti na nás číhá spousta nebezpečí. Existují tisíce aplikací, které mohou být teoreticky i prakticky zneužity. Některá nebezpečí lze poměrně snadno eliminovat například dodržováním elementárních bezpečnostních zásad, jako jsou silná hesla nebo aktuální verze softwaru. Existují však i ohrožení, která se eliminují poměrně těžko, a jejich odhalení není vůbec snadné. O těchto nebezpečích bude dnes řeč.

Většinou jde o útoky, které využívají samotného protokolu TCP/IP. Tento protokol je starý přes 30 let a postupem doby se ukázalo, že ne vše je navrženo a vymyšleno správně. Musíme však brát v potaz, v jakých podmínkách byl protokol TCP/IP navržen a stvořen. Při jeho vývoji se ani zdaleka nepočítalo s tím, že za několik desítek let bude počítačová síť celosvětová, se stamiliony uživatelů. Ale nyní již k samotným nebezpečím. Je však také třeba zdůraznit, že při rozumné míře obezřetnosti a dodržení některých pravidel lze eliminovat i tato ohrožení.

Sledování síťového provozu

Pro sledování síťového provozu se vžilo anglické označení sniffing (česky "čmuchání"). V podstatě jde o zachytávání a analýzu síťového provozu. My se nyní podíváme na princip a podmínky, za kterých je tento druh útok možný. Především je třeba zdůraznit, že se tento typ útoků týká v převážné většině sítí LAN. Další podmínkou je, aby síť nepoužívala přepínané spojení (toto pravidlo se dá obejít, ale k tomu až za chvíli). Nyní si ukážeme, jak v takové síti LAN probíhá komunikace. Všechna data, bez ohledu na to, komu nebo od koho jsou určena, procházejí přenosovým médiem (kabelem) v seskupení nazvaném rámec. Každý rámec je určen konkrétní MAC adrese, což je adresa každé konkrétní síťové karty, jež je u většiny karet neměnná a byla kartě přidělena výrobcem. Každá karta na stejném segmentu sítě přijme každý rámec, který přenosovým médiem prochází a zjistí, zda je určen pro ni. Pokud ano, předá tento rámec ke zpracování vyšší vrstvě. Pokud ne, je rámec ignorován. Může nastat ještě jedna situace, a sice taková, kdy je rámec určen každé MAC adrese (broadcast).

Dejme tomu tedy, že se ze svého počítače, který je součástí nějaké LAN, připojujete k místnímu poštovnímu serveru. Zadáte uživatelské jméno, heslo, bez problémů si

stáhnete poštu. V tu chvíli vás ani nenapadne, že by vaše heslo nebo vaši poštu mohl číst kolega vedle u stolu nebo někdo na druhé straně zeměkoule. Možné to však je. Zmíněný kolega to mohl provést naprosto lehce. Existují totiž programy, které umožňují čtení rámců i v případě, že jsou určeny pro jinou MAC adresu. Takové programy se nazývají sniffery. Stačí takový program nainstalovat a spustit, zbytek již program provede sám. Pro zdatného programátora by nebylo napsání podobného programu vůbec těžké. Program přepne síťovou kartu do takzvaného "promiskuitního" modu a sleduje a analyzuje veškerý provoz v daném segmentu sítě. Existují jednoduché sniffery, jako je například unixový Tcpdump, až po velmi sofistikované, jako je třeba Hunt. Obrana proti Huntu se tedy nabízí sama šifrování. Pokud provoz šifrujeme, budou útočníkovi veškerá data k ničemu, neboť je nebude moci přečíst. Nejvhodnější alternativou jsou zřejmě SSH a SSL. Proto je vhodné nahradit stávající služby, které používají textová hesla. Musíme ale také vědět, kdo se pokouší provoz sledovat. Pokud je to někdo z naší lokální sítě, obrana je celkem obtížná. Existují však programy, jež dokáží nalézt v síti karty, které jsou v promiskuitním modu.

Sniffer však na některém počítači vaší LAN může nainstalovat i nějaký vetřelec z venku. Pokud se mu podaří nabourat se do vaší sítě, pravděpodobně tak i učiní, neboť není snazší cesty jak získat velké množství různých uživatelských jmen a hesel. Nejlepší obranou zde je zabezpečit vaši síť tak, aby se do ní nikdo nedostal a nemohl tak instalovat sniffer.

DNS

DNS (Domain Name Service) je jedna z nejdůležitějších služeb, které můžeme v prostředí sítí nalézt. Tato služba se stará o převod doménových jmen na IP adresy a opačně. Tato služba denně usnadňuje život milionům uživatelů (některým i nevědomky :). Každý objekt (server, router), který je součástí nějaké sítě založené na IP (intranet, internet), má jedinečnou identifikaci. Touto identifikací je jeho IP adresa. Jde o číslo ve tvaru xxx.xxx.xxx.xxx. Například tedy 192.168.1.1. Toto číslo je jedinečné a nemohou ho v jednu chvíli sdílet dva objekty připojené do sítě. Tento jednoznačný identifikátor má však i svou jmennou alternativu, například http://www.firma.cz. A právě o převod mezi těmito dvěma identifikátory se stará DNS. Jistě je pohodlnější pamatovat si adresy jako http://www.pcworld.cz nebo http://www.google.com než jakousi "směsici" čísel. Výklad fungování DNS ale není v rámci našeho článku, proto se podíváme, jak se dá DNS zneužít.

Falšování DNS

Falšování DNS (DNS spoofing) je poměrně nebezpečné. K jeho provedení je však třeba splnit několik podmínek. Předně je třeba, aby útočník mohl sledovat váš síťový provoz (viz výše). Pokud má přístup do vaší LAN, může na svém (nebo ovládnutém) stroji spustit program, který odchytává všechny DNS dotazy a snaží se na ně odpovídat podle útočníkova záměru. Cílem je tedy podvrhnout DNS odpověď, a přesměrovat tak hosta na jiný systém. Útočník má například někde na síti server, který je přesnou kopií nějakého, dejme tomu webového e-mailového serveru. Na první pohled nerozeznatelný od

originálu. Jeho cílem je přesměrovat všechny uživatele vaší sítě právě na tento server. Pokud tedy nějaký uživatel zadá ve svém prohlížeči adresu http://www.webmail.něco, jenž má ve skutečnosti adresu 192.168.1.1 (tuto adresu mít ve skutečnosti nemůže, neboť jde o adresu z bloku privátních adres určených k použití na sítích nepřipojených přímo do internetu, nicméně jako ilustrace stačí), pokusí se útočníkův program podvrhnout odpověď DNS serveru a odpoví klientovi tak, že server http://www.webmail.něco má adresu 192.168.1.100. Pokud tato odpověď dorazí k uživatelovo systému dříve, něž odpověď skutečného DNS serveru, má útočník vyhráno. Jak se tedy proti tomuto druhu útoku bránit? Prvním a univerzálním pravidlem je opět nepouštět útočníka do sítě. Pokud nebude mít přístup do sítě, nezmůže nic. Chová-li se takto některý z legitimních uživatelů sítě, je řešením vyhledávač snifferů. Existuje i další možnost a tou je použití DNS serveru, který podepisuje své odpovědi.

Další zneužití DNS

Existuje ještě jeden druh útoku, jímž lze zneužít DNS server. Tento druh útoku je však poměrně zastaralý a je účinný jen proti starším verzím DNS serveru BIND (unixový DNS server). Nicméně někde se se staršími verzemi tohoto programu stále ještě setkáváme, takže jen v krátkosti. Tento druh útoku se nazývá Cache poisoning (otrava cache) a v principu jde o podvržení nějakých legitimních odpovědí DNS. K tomuto triku nemusí mít útočník přístup do vaší sítě. Řešení je velmi prosté, používejte aktuální software.

Směrování a přeposílání

Směrování je vlastnost IP protokolu, která dovoluje určit, kudy se budou pakety na své cestě sítí ubírat. Podmínkou však je, aby systémy, jež útočník uvede ve své cestě, umožňovaly směrování. Pomocí směrování lze velmi snadno falšovat pakety a získávat tak neautorizované informace. Řešením je vypnutí směrování (Windows je standardně neumožňují, většina výchozích instalací Linuxu také ne). Pokud směrování potřebujete, nastavte dobře ACL na všech vašich směrovačích.

Směrování naopak umožňuje útočníkovi přístup do vnitřní sítě, která může být připojena pomocí jednoho systému a z okolního internetu nemusí být vůbec viditelná. Podmínkou k tomuto útoku jsou špatně nastavená přeposílací pravidla, proto vždy pečlivě tato pravidla zkontrolujte a otestujte.

Unášení relací

Unášení relací patří mezi pokročilejší metody útoku. Od útočníka vyžaduje jistou dávku znalostí síťové komunikace. O to je tento druh útoku nebezpečnější. Podmínkou je opět, aby útočník mohl sledovat síťový provoz. Ideálním nástrojem k unášení relací je již několikrát zmiňovaný Hunt. Útočník spustí na svém stroji program Hunt a začne sledovat komunikaci mezi dvěma systémy. Usoudí-li, že nastala vhodná chvíle k převzetí kontroly nad spojením, pokusí se ji převzít. Pokud se mu to podaří, má nad spojením plnou moc a cílový systém nezjistí, že komunikuje s někým jiným. Útočníkovi je tedy dovoleno dělat vše, co mohl dělat původní vlastník spojení. Obrana spočívá opět v použití šifrování a dobrém zabezpečení sítě.

Prostředník (Man in the middle)

Tyto útoky patří mezi méně sofistikované a lze se proti nim velmi lehko bránit. Spoléhají totiž na uživatelskou netečnost a nedůslednost. Tyto druhy útoků lze aplikovat i na šifrované protokoly jako SSH nebo SSL. Nejde však o chybu těchto protokolů, jen o zneužití důvěry uživatelů.

SSH

Jak již vyplývá z názvu tohoto útoku, je principem udělat prostředníka ve spojení. Útočník potřebuje zachytit úvodní spojení od klienta, poté se připojit ke skutečném cílovému serveru a předstírat, že je koncový systém. Klient nepozná, že komunikuje prostřednictvím třetího systému. Až na jednu, dost podstatnou maličkost. Každý SSH server má svůj vlastní identifikační klíč. A jelikož útočník tento klíč ve většině případů nemá, potřebuje tuto ochranu nějak obejít. Nejčastěji učiní to, že si vytvoří klíč vlastní. To má ale za následek, že pokud se klient k systému útočníka připojí, zobrazí se mu varování o změně klíče. Tento útok tedy spoléhá na uživatelovu netečnost. Pokud uživatel slepě odsouhlasí (nebo odkliká :)) všechna varování, nic už útočníkovi nebrání ve sledování provozu. Pokud se vám tedy někdy objeví podobná hláška, zbystřete a kontaktujte správce serveru.

SSL

Útok s pomocí prostředníka lze aplikovat i na protokol SSL. Princip je naprosto stejný. Tento druh útoku je však ze subjektivního hlediska ještě více nebezpečný, neboť protokol SSL se používá v prostředí webu, kde se pohybuje i velké množství nezkušených uživatelů, jež jsou z některých obrázkových operačních systémů zvyklí slepě odkliknout každé okno, které se objeví. Proto je důležité kontrolovat certifikáty SSL, zdali jsou podepsány oficiální certifikační autoritou. Pozor by si měli uživatelé dávat zejména na samopodepsané certifikáty.

Hesla

I když je vaše síť dokonale zabezpečena, uživatelé dodržují bezpečnostní zásady a vše pečlivě kontrolujete, sledujete a analyzujete, stále ještě existuje způsob, jak vaši síť napadnout. Většina serverů nabízí nějaké služby. Bez nich by internet nebyl tím, čím je dnes. Přístup k těmto službám může být nabízen přes síť a rozdílné skupině uživatelů. Některé služby můžeme zpřístupnit všem, jiné jen někomu. Bez ohledu na to, komu službu nabízíme, vždy na ni může být proveden útok přes síť. Útočník jistě zkusí spoustu jiných metod a triků, ale pokud se ukáže, že je síť zabezpečena opravdu dobře a většina tradičních útoků nepomůže, může se rozhodnout zaútočit na vaše hesla. Tento útok je poměrně časově náročný a výsledky nejsou vůbec jisté, nicméně někdy je to poslední metoda, jak se dostat dovnitř. Zaútočit na hesla lze na téměř všechny protokoly, jež poskytují nějakou formu autentizace. Může jít například o SMTP, POP, FTP, SSH a další. K tomuto útoku existují také desítky nástrojů. My se nyní podíváme na různé metody hádání hesel a povíme si, jak se proti nim bránit.

Slovníkový útok

Slovníkový útok je nejpoužívanější metoda při hádání hesel. Základem úspěchu je rychlý procesor, optimalizovaný program a rozsáhlý wordlist. Pomocí vhodného programu lze

útočit téměř na všechna hesla. Při hádání hesel však nemusí jít jen o útoky přes síť. Hesla mohou být hádána i lokálně. To záleží na požadavcích a možnostech. Principem je tedy vzít nějaké slovo, upravit ho do vhodného tvaru (v závislosti na algoritmu) a porovnat jej s originálním heslem. U síťových služeb je cílem toto heslo odeslat po síti (v jaké formě, záleží na protokolu). Při slovníkovém útoku se jako potencionální hesla vybírají slova, která jsou uložena v souboru. Nezkoušejí se tedy různé kombinace znaků. Některé slovníky jsou opravdu rozsáhlé. Proto je dobré zvolit si heslo, které v takovém slovníku obsaženo nebude.

Tvorba hesla

Zkuste si zvolit nějaké libovolné heslo a někam si je napište. Provedeme test, zdali jste heslo zvolili správně. Nyní budeme potřebovat slovník. Doporučuji stáhnout si slovníky z adresy http://www.phreak.org/html/wordlists.shtml. Jde o velmi rozsáhlou sbírku slovníků z nejrůznějších jazyků. Můžete si stáhnout slovníky podle témat nebo jazyků. Doporučuji stáhnout všechny. Nyní je načase zjistit, zda je vámi zvolené heslo součástí některého z nich. V UNIXu můžete použít například grep mojeheslo slovnik. Zde je několik rad, jak vytvořit dobré heslo.

Heslo by mělo obsahovat:

- malé znaky anglické abecedy a-z
- velké znaky anglické abecedy A-Z
- číslice 0-9
- interpunkční znaky *, #, @,] atd.
- minimálně 6 znaků

Heslo by nemělo obsahovat:

- jakékoliv slovo
- jakékoliv jméno
- kombinaci jména (slov) a číslic, například honza52
- údaje o vaší osobě přezdívka, jméno manželky atd.
- čísla vztahující se k někomu nebo něčemu
- datum narození kohokoliv, telefonní čísla, adresy atd.

Je nutno brát zřetel na to, že byste neměli jedno heslo použít dvakrát. Pokud tedy máte zřízeno několik e-mailových adres, měli byste u každé z nich použít jiné heslo. Mohlo by se stát, že by bylo vaše heslo prozrazeno (například pomocí snifferu), a útočníkovi byste tak poskytli přístup do mnoha jiných systémů.

14. Literatura

Habraken, Joseph W. Průvodce úplného začátečníka pro Počítačové sítě : není zapotřebí žádných předchozích zkušeností!. 1. vyd. Praha : Grada, 2006. ISBN 80-247-1422-1.

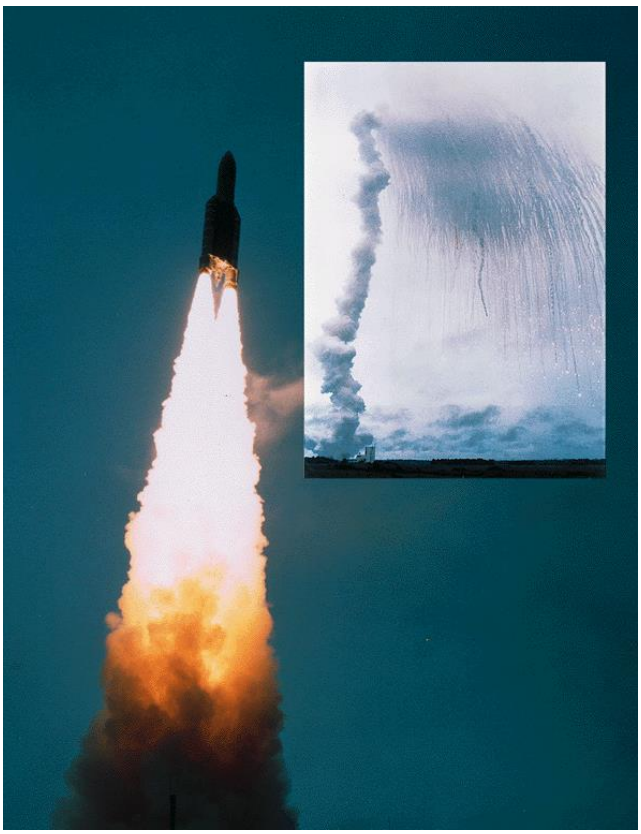
STALLINGS, William. *Local and metropolitan area networks*. 6th ed. Upper Saddle River: Prentice Hall, 2000. xvi, 478 s. ISBN 0-13-012939-0.

PUŽMANOVÁ, Rita. Moderní komunikační sítě od A do Z : [technologie pro datovou, hlasovou i multimediální komunikaci]. 2. aktualiz. vyd. Brno: Computer Press, 2006. 430 s. ISBN 8025112780.

THOMAS, Robert M. *Lokální počítačové sítě*. Vyd. 1. Praha: Computer Press, 1996. 277 s. ISBN 80-85896-45-1.

SOFTWAROVÉ INŽENÝRSTVÍ

1. Úvod do softwarového inženýrství



Chyba v software může způsobit pád aplikace v telefonu, a nebo nějaký opravdu velký průšvih. Některé reálné důsledky nepovedeného software uvádí Richta (2011) ve své přednášce jde například o:

Pád rakety Ariane 5: Díky chybě dal řídicí počítač rakety příkaz k současnému vychýlení trysek urychlovacích bloků, tak i trysky motoru. Tím se kurs rakety prudce změnil a v důsledku aerodynamických sil se horní část rakety odlomila. Byl aktivován vlastní autodestrukční systém rakety a raketa se změnila v oblak hořících úlomků. Celková cena: 100 mil. $ (včetně družic Cluster, které nesla, 500 mil. $).

Výpadek elektřiny USA (2003): Způsobena neregistrovaným výpadkem automatizovaného hlásiče poruch v elektrárně u Niagary, který se kaskádně rozšířil sítí. Výpadek vyřadil z provozu celkem 21 elektráren. Postiženo bylo až 50 mil. obyvatel, zemřeli 3 lidé, škoda 6 mld. $

Havárie přistávacího modulu na Marsu (1999) Problém komunikace mezi komponentami – uživatel rozhraní očekával hodnotu v kilometrech, poskytovatel ji udával v mílích. Namísto plánovaných 140 až 150 kilometrů tedy zamířila pouze 57 kilometrů nad povrch. V té výšce je však atmosféra Marsu na sondu příliš hustá. Climate Orbiter shořel nejspíše ve výšce kolem 80 kilometrů.
(Zdroj: Technet.cz)

1.1. Důvody vzniku softwarového inženýrství

Na začátku vývoje počítačů ve 40tých a 50tých letech byl jejich výpočetní výkon velmi nízký. Tento výkon se časem zvyšoval geometrickou řadou podle známého Moorova

zákona. Vývoj software byl v té době v plenkách. Nebyli známi žádné postupy, které vývoj systematizovali. V šedesátých letech při dalším zvýšení výkonu počítačů se pomalu začíná mluvit o softwarové krizi.

Charakteristickými znaky softwarové krize bylo neúnosné prodlužování a prodražování projektů, nízká kvalita programů, nesnadnost či nemožnost údržby a inovace, špatná produktivita práce programátorů, neefektivita vývoje, nejistota výsledku a řada dalších.

Příčin této krize bylo hned několik:

- Špatná komunikace mezi osobami tvořícími software a také mezi vývojáři a zákazníkem.
- Nesprávný přístup. Požadovaný software vyhotovoval a schvaloval vývojář a nespokojený zákazník byl označen za osobu, která tomu nerozumí.
- Špatné plánování celého projektu. Vývojáři předpokládali, že to nějak stihnou.
- Nesprávné odhady trvání vývoje, nákladů, rozsahu.
- Nízká produktivita práce. Programátoři se zabývali vším možným, jen ne dosti tím, čím měli.
- Neznalost základních pravidel. Např. Brooksův zákon z roku 1975: „Přidání řešitelské kapacity u zpožděného softwarového projektu způsobí jeho další zpoždění.“
- Podcenění hrozeb a rizik. Málo byly sledovány hrozby a místo snadného předcházení přerůstaly ve velké problémy.
- Nezvládnuté technologie. Falešná představa, že po zavedení nové technologie se potíže samy vyřeší.

Jako reakce byla uspořádána v roce 1969 konference, která zavedla základní principy softwarového inženýrství.

V té době byl definován takto:

Softwarové inženýrství je disciplína, která se zabývá zavedením a používáním řádných inženýrských principů do tvorby software tak, abychom dosáhli ekonomické tvorby software, který je spolehlivý a pracuje účinně na dostupných výpočetních prostředcích.“

Softwarové inženýrství bychom mohli popsat jako inženýrskou disciplínu, zabývající se reálnými problémy vývoje rozsáhlých softwarových systémů. Tato disciplína v sobě obsahuje několik různých myšlenkových postupů. Je to inženýrská disciplína, díky používání pragmatických postupů. Mezi které patří znalosti z oblasti specifikace požadavků na softwarový produkt, jeho analýzy, návrh, implementaci a testování a samotné zavedení systému u koncového zákazníka. Zároveň v sobě integruje manažerský přístup vedení projektu umožňující efektivní využití jednotlivých postupů, jejichž hlavním cílem je efektivně vytvořit kvalitní produkt software.

1.2. Psychologická odbočka

Kognitivní psychologie popsala obecné rozdíly v řešení problémů mezi experty na danou oblast a nováčky. Experti si dokáží vybavit částečná řešení podobných situací a použít předchozí znalosti ke klasifikaci a definici problémů (Eysenck a Keane, 2008). Experti řeší zadaný problém tak, že vezmou zadání problému, všechny vnější okolnosti a na základě těchto informací se snaží najít řešení. Nováčci se naopak nejdříve snaží vytipovat řešení, a poté se snaží najít ověření, zda lze ze známých faktů vyvodit jimi nalezené řešení. Používají tedy strategii zpětného posunu (Nolen Hoeksema at al., 2012). Dle Plháková je největší rozdíl mezi nováčky a experty v rozsahu organizaci znalostí (Plháková, 2003). Mezi znaky úspěšných jedinců (expertů) v odborných profesích patří velmi rozvinuté auto regulativní dovednosti (popisované v samostatné kapitole) umožňující efektivní uplatňování dovedností a vědomostí (Helus & Pavelková, 1992). Porozumění problémům bývá u laiků relativně povrchní. Znalci nejprve odhadnou podstatu problému a snaží se ji ujasnit a utřídit. Nepouštějí se do řešení, dokud nemají připraven efektivní plán. Odborníci věnují přípravě a plánování proporcionálně více času než laici, ale celkové řešení problému jim trvá kratší dobu (Plháková, 2003). Říčan (2016) experty popisuje: „ve své oblasti disponují nejenom větším objemem znalostí (kvantitativní stránka), ale rovněž se odlišují kvalitativně, a to ve smyslu více rozvinutého strategického chování během řešení úkolových situací (odlišují se tedy v jejich schématech v učení). Experti mají více a lépe organizované doménově-specifické znalosti a tím jsou schopni rychleji získat nové informace ze své oblasti erudice, jelikož předchozí znalosti, které jsou k dispozici, fungují jako integrační struktura pro nově příchozí informace prostřednictvím asociativního pojení pouze s minimem kognitivního úsilí.“

Obecně, tak lze říci, že softwarové inženýrství v sobě integruje strategie řešení problémů, které podle kognitivní psychologie náleží expertům.

2. Životní cyklus informačních systémů

Velmi trefné přirovnání používá Wikipedie, kdy životní cyklus přirovnává k mostu. Most je ve své podstatě prostředek k zjednodušení nějaké lidské aktivity. Podobný smysl by měly plnit i softwarové systémy. Stejně jako mosty se software musí navrhnout, používat a udržovat a na konci své životnosti pak demontovat. Jistě ve stavebnictví existuje životní cyklus staveb, který bude dosti podobný životnímu cyklu software.

Existuje velké množství klasifikací životního cyklu softwarových systémů. My jsme zvolili Systems Development Life Cycle(SDLC) od The Department of Justice USA z roku 2003.

- **Inicializace** (Initiation) – nejdříve je třeba, aby objekt mající dostatek financí (zákazník) měl potřebu nějakého zlepšení realizovaného pomocí software. Nazývejme jej záměrem.
- **Vývoj konceptu** (System Concept Development) - Tento záměr je přezkoumán z hlediska proveditelnosti a vhodnosti. Je na hrubo popsán rozsah systému na jehož základě jsou vyčísleny náklady, které musí být schváleny zákazníkem.
- **Plánování** (Planning Phase) - Tento koncept je dále rozvíjen tak, aby popsal, jak bude podnik fungovat po zavedení schváleného systému. Jsou naplánovány a schváleny konkrétní plány projektu.
- **Analýza požadavků** (Requirements Analysis Phase) - Všechny požadavky jsou definovány na úroveň detailů, která postačuje pro pokračování návrhu systému. Typicky vymezují požadavky z hlediska dat, výkonu systému, zabezpečení a požadavků na údržbu systému a podobně.
- **Design** (Design Phase) Fyzické charakteristiky systému jsou navrženy v průběhu této fáze. Jsou definovány hlavní části systému a jejich vstupy a výstupy. Vše, co vyžaduje vstup nebo schválení uživatele, musí být dokumentováno a přezkoumáno uživatelem.
- **Vývoj** (Development Phase) veškeré specifikace vytvořené v předchozích fázích jsou vloženy do vnikajícího software. Následně jsou testovány.
- **Integrační a testovací fáze** (Integration and Test Phase) Jednotlivé komponenty systému jsou složeny a systematicky testovány.
- **Implementační fáze** – systéme je nainstalován a zprovozněn u zákazníka. Zákazník si musí systém vyzkoušet a odsouhlasit. Fáze končí nasazením produktu do běžného provozu.
- **Provoz a údržba** (Operations and Maintenance Phase) Provoz systému by měl být monitorován pro odpovídající službu. V případě potřeby jsou zapracovány potřebné úpravy systému. Tento proces pokračuje tak dlouho, dokud systém může být účinně přizpůsobován potřebám zákazníka. Pokud proces již nelze dále přizpůsobovat anebo je to již příliš drahé, dochází k fázi plánovaní nového software.

- **Ukončení používání** (Disposition Phase) – Ukončení používání systému. Zvláštní důraz je kladen na řádné zachování dat zpracovávaných systémem, aby se údaje mohly efektivně přenést do jiného systému nebo archivovat v souladu s platnými předpisy a politikami správy záznamů pro případný budoucí přístup.

V tomto textu si podrobně popíšeme celý životní cyklus software. Ukážeme si jednotlivé metody vývoje software, abychom se následně podrobně zaměřili na metodu RUP (Rational Unified Process). Tato metoda podrobně pokrývá z výše uvedených od fáze Inicializace až po fázi implementační. Metoda RUP hodně pracuje s UML, proto si ukážeme celkem šest různých druhů diagram UML. Seznámíme se s modelováním firemních procesů (Business Process Model and Notation), které může být užitečné na začátku tvorby nového softwarového systému. Samostatně si pak popíšeme testování software a jeho údržbu pomocí metodiky ITIL.

3. Metody vývoje softwaru

Je v obecné rovině souhrn postupů, pravidel a nástrojů vedoucích k tvorbě software. Občas je též nazývaný softwarový proces. Jednotlivé nástroje určené pro tvorbu software se nazývají Framework. Cílem těchto postupů je efektivní vývoj a udržování software.

Využívání těchto postupů by mělo snížit riziko nepředpokládaných problémů a navyšuje přehlednost harmonogramu práce na daném software. Pokud jednotlivý vývojáři využívají stejné postupy. Výrazně se jim usnadní spolupráce díky dopředu daným pravidlům vývoje programu.

Postupem času vzniklo velké množství metod softwarového vývoje. Dodnes však neexistuje detailně a přesně definovaná podoba metody vývoje software, který by mohl přijat jako referenční. Patří mezi ně například:

3.1. Vodopádný přístup

je postupný vývojový proces, ve kterém je na vývoj pohlíženo, jako postupnou proces, kdy se postupně prochází jednotlivé jeho části: návrh, implementace, testování, integrace a údržba, případně předávání. Tento model vznikl již v sedmdesátých letech. A dá se označit, jako základní model, od kterého se odráží ostatní modely.

Přechod z jedné fáze do druhé se realizuje až v momentě částečného, nebo úplného dokončení předchozí fáze. V reálném životě ale bohužel často tento model není možné využít. Například specifikace programu často zákazník mění až v průběhu tvorby tohoto programu anebo až při jeho využívání. Jako problém tohoto modelu se dají označit:

- Není možné v průběhu vývoje software odhadnout výslednou kvalitu produktu, na základě splnění předem daných kritérií na specifikaci software
- Výsledný produkt záleží na kvalitě zadání na výsledný software
- Čas potřebný k vývoji software je příliš dlouhý

Snaha o opravení nedostatků v tomto přístupu vedla ke vzniku dalších přístupů. Tyto procesy popisuje prof. Vondrák (2002):

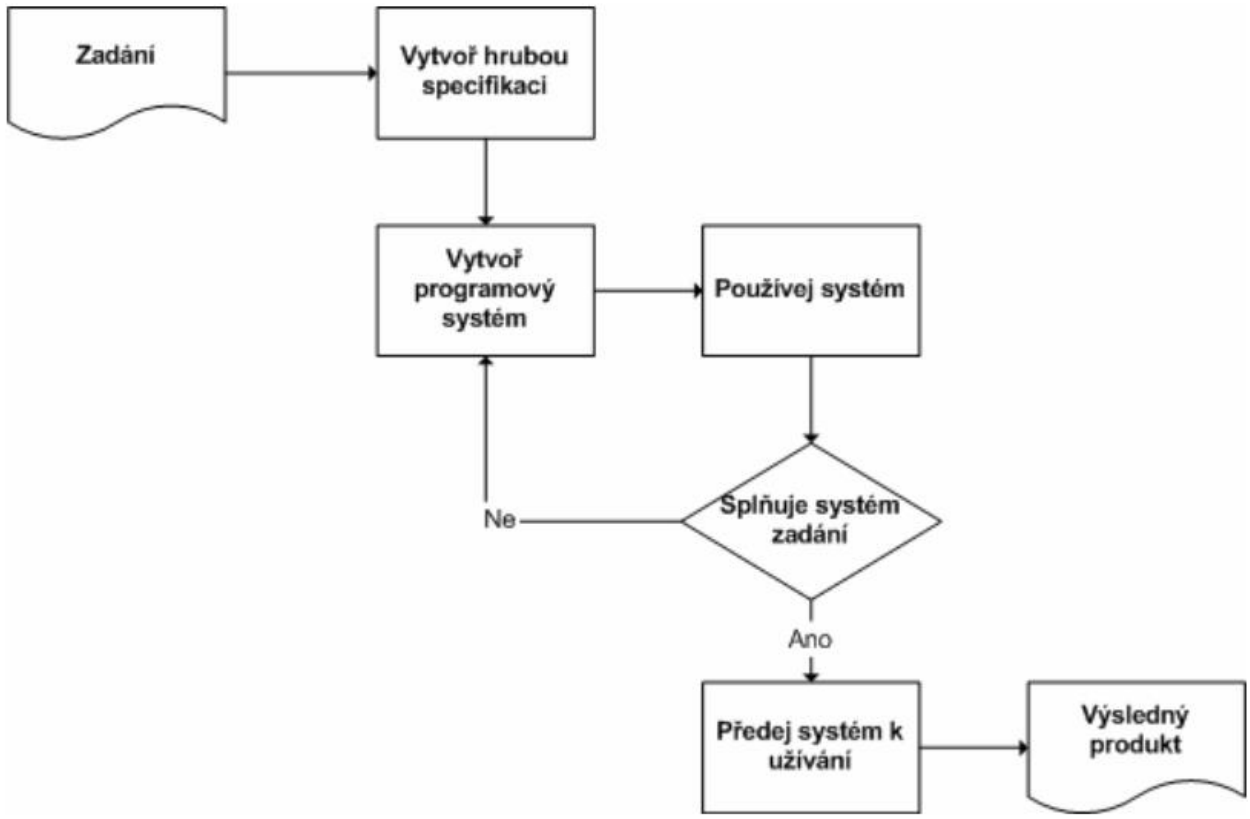
3.2. Iterativní přístup

Jednotlivé aktivity vývoje se opakovaně vykonávají v „iteracích“. Kdy se v každé iteraci přidá do vznikajícího software vybraná malá skupina funkcí vedoucích k cílovému stavu. Pomalu tak vyrůstá požadovaný program a zpřesňuje se jeho funkcionalita. Zákazník tak má možnost vyzkoušet si vyvíjený software již v rané fázi vývoje. Může tak rychleji upřesňovat své požadavky na vyvíjený software. Případné chyby, nedodělky, či chybně zadané požadavky na program se tak odhalí výrazně dříve. Díky tomu, že zákazník vidí

software již od raného vývoje, může více ovlivňovat k obrazu svému výslednou podobu produktu. V tomto přístupu se poněkud smazávají rozdíly mezi jednotlivými fázemi, tak jak o nich mluví vodopádný přístup. Spíše se nabízí paralela, kdy každá iterace je jeden malý vodopád. Do této kategorie spadá i proces RUP, kterému se budeme věnovat v samostatné kapitole.

„průzkumné“ programování

prof. Vondrák (2002) ještě zmiňuje, jako odstrašující model, který je občas využíván v praxi. Tento model nazývá průzkumné programování.



3.3. Agilní techniky

Většina softwarových procesů historicky vznikla ve velkých korporacích. Tyto postupy vyžadovali striktní dodržování naplánovaného postupu a povinné vytváření celé řady dokumentů. Což je v pořádku v případě velkých firem, kdy na projektu pracuje velká spousta lidí, nebo je třeba vysoký stupeň spolehlivosti (například řízení rozvodné elektrické sítě). V případě menších projektů ale čas strávený dodržováním všech postupů a jiných formalit byl mnohdy delší než čas určení pro programování a testování.

Jako reakce tak vzniká na začátku nového milénia proces Agilní metody, či techniky. Základní principy těchto technik byly v roce 2001 formulovány v manifestu agilního vývoje software.

- Lidé a jejich interakce jsou důležitější než procesy a nástroje

- Fungující software je důležitější než podrobná dokumentace
- Spolupráce se zákazníkem je důležitější než uzavřené smlouvy
- Reagování na změny je důležitější než dodržování plánu

Hlavní idea těchto metod je umožnit vývojářům flexibilní nesvazující přístup k tvorbě software. Vývojář se zaměřuje na vývoj software, jeho navrhování a dokumentování již tolik neřeší. Tyto metody se řadí mezi iterativní metody. Je založen na aktivní spolupráci se zákazníkem. Rozpis práce se většinou řeší v horizontu další iterace. Ze své trošku trochu „anarchistické“ povahy jsou tyto techniky vhodné pro menší kvalitní vývojářské týmy. Na členy týmu jsou kladeny zvýšené nároky v oblasti komunikace a spolupráce.

V současné době existuje několik agilních metod. Mezi nejznámější patří extrémní programování a Scrum (Sklenář, 2007)

4. RUP

Proces RUP (Rational Unified Process) vnikal v druhé polovině devadesátých let ve spolupráci několika firem pod vedením firmy Rational. Tato firma se pak v roce 2003 stala divizí IBM. RUP není jediným konkrétním předepsaným procesem, nýbrž adaptivním iteračním procesním rámcem, který má být přizpůsoben potřebám vývojových organizací a týmů vyvíjejících software. Toto přizpůsobení je realizováno tak, že vývojové týmy si vybírají prvky procesu, které odpovídají jejich potřebám. RUP vychází z UP (inified Process). RUP patří v současné době mezi výrazně rozšířené modely softwarového procesu. Podporuje jej mnoho nástrojů.

V klasickém vodopádném přístupu jsou běžně rozděleny jednotlivé role. Stává se tak, že analytik od zákazníka přebíral představu fungování systému. Tuto představu následně přepíše do dokumentu, který předá programátorovi. Pokud dochází k absenci pravidelné spolupráce mezi analytikem a vývojářem. Vývojář si musí „dovozovat" požadavky na systém z požadavků sepsaných v dokumentu. Velmi jednoduše si pak může tyto požadavky vyhodnotit jinak, než zamýšlel zákazník.

Tento proces je postaven na několika zásadách. Tento proces je postaven na ideji **iteračního vývoje** software. Díky tomu by měl být každý program vyvíjen ve verzích (iteracích) Každá tato iterace by měla být vytvořena spustitelným kódem.

Žáček (2017) uvádí základní principy RUP:

- Snahu minimalizovat rizika projektu co nejdříve a neustále.
- Ujištění se, že dodáváme zákazníkovi přidanou hodnotu.
- Zaměření na spustitelný software.
- Zapracovat změny v časných fázích projektu.
- Brzké nastínění spustitelné architektury.
- Znovupoužití existujících komponent.
- Úzká spolupráce, všichni jsou jeden tým.
- Kvalita je způsob provádění celého projektu, nejen část (testování).

Prvotní specifikace na software je zásadním předpokladem pro kvalitní vývoj produktu. RUP vytváří **Systém správy požadavků.** Od jejich získání, dokumentaci, po monitorování změn v požadavcích. Tento systém následně tvůrci software v komunikaci se zákazníkem. Stejně tak jsou řízeny změny ve způsobu vývoje software.

Mnohé části softwarových produktů jsou si podobné. Proto je ztráta času je znovu vyvíjet. RUP zavádí **vývoj na základě komponent**. V momentě, kdy máme k dispozici potřebné komponenty, vývoj produktu se stává skládáním jednotlivých komponent dohromady. Toto skládání můžeme připodobnit například k legu, anebo ke skládání stolního počítače.

Díky komplexnosti vyvíjených programů je pro lepší představení důležitá **vizualizace softwarového systému**. RUP pracuje s jazykem UML.

Pro zajištění spokojenosti zákazníka díky funkčnímu software a pro potřeby zpětné vazby pro vývojáře je důležité **ověření kvality vytvářeného software**. Toto ověření by se mělo týkat všech vývojářů produktů. RUP specifikuje specifické postupy popisující kvalitu vytvářeného software.

Tvorba software je v RUP rozdělena celkem do čtyř fází. Fáze v RUP bychom mohli pojmenovat jako jednotlivé statusy projektu, jeho změn v čase. Každá fáze často obsahuje několik iterací.

4.1. Schématický model

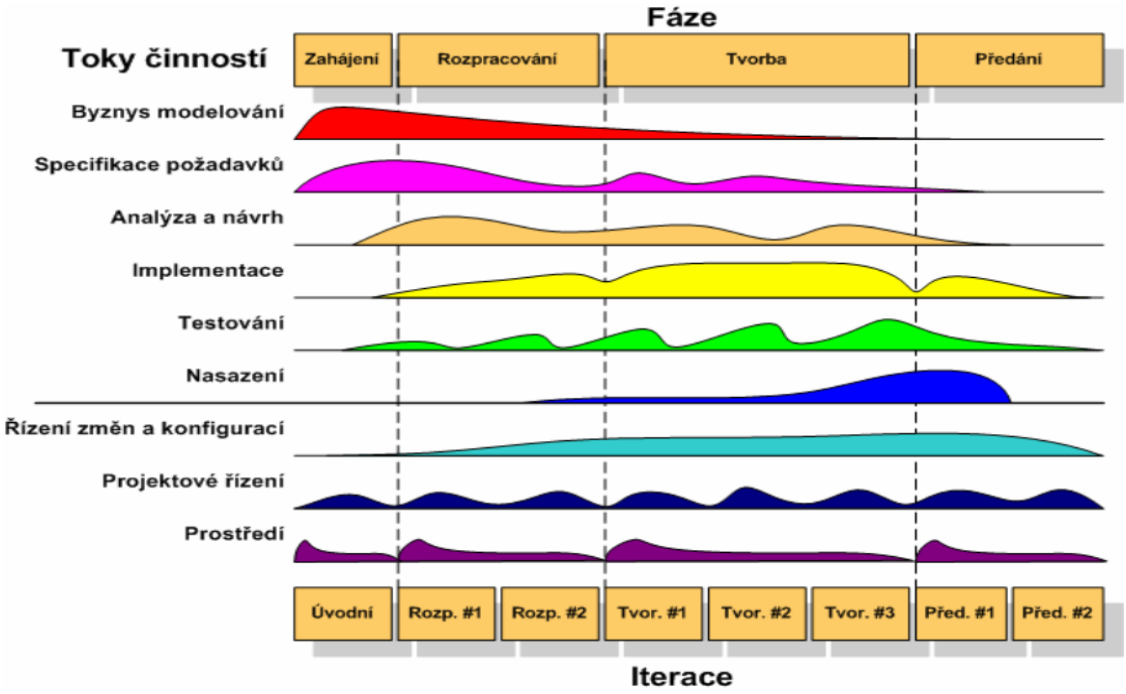
Samotný schématický model se sestává u celkem šesti inženýrských disciplín a tří. Mezi inženýrské disciplíny patří:

- Byznys modelování
- Specifikování požadavků
- Analýza a návrh
- Implementace
- Testování
- Nasazení

Do pomocných disciplín spadá:

- Správa konfigurací a změn
- Řízení projektu
- Příprava prostředí pro projekt

Přibližnou četnost aktivit u jednotlivých disciplín v závislosti na jednotlivých fázích a iteracích projektu vypracoval prof. Vondrák (2002).



Přibližné časové a zdrojové rozdělení je zobrazeno vypadá asi takto:

	Zahájení	Rozpracování	Tvorba	Předání
Zdroje	Cca 5%	20%	65%	10%
Čas	10%	30%	50%	10%

Žáček (2017) uvádí, jaké jsou výstupy jednotlivých fází:

- Výstupem Inception (zahájení) je pochopení problematiky, vize projektu, identifikovaná rizika.
- Výstupem Elaboration (rozpracování) je spustitelná, otestovaná architektura (=
- fungující část aplikace).
- Výstupem Construction (tvorba) je beta-release aplikace, relativně stabilní, opět spustitelná, téměř kompletní aplikace.
- Výstupem Transition (předání) je pak již produkt připravený k finálnímu nasazení včetně veškeré dokumentace a hardware. Zásadní rozdíl je také v tom, že každá fáze může

Na jednotlivých fázích se běžně podílí různé počty zaměstnanců a různých pozic. Je ale důležité, aby tito zaměstnanci stále pracovali spolu v jednom či více týmech. Pokud je to potřeba jsou si navzájem k dispozici k dovysvětlení nejasností.

4.2. Zahájení (Inception)

Na začátků je třeba stanovit rozsah projektu, posoudit podmínky, rizika, časové a finanční náklady realizace celého projektu. Žáček (2017) uvádí celkem 5 cílů této fáze:

- Porozumění tomu, co vytvořit – vytvoření vize, definice rozsahu systému, jeho hranic; definice toho, kdo chce vytvářený systém a co mu to přinese.
- Identifikace klíčových funkcionalit systému – identifikace nejkritičtějších Use Casů.
- Návrh alespoň jednoho možného řešení (architektury), (tedy zda je vůbec možné pokračovat ve vývoji pozn. autor)
- Srozumění s náklady, plánem projektu, riziky.
- Definice/úprava procesu, výběr a nastavení nástrojů.

Definice a význam požadavků

V rámci zjišťování požadavků se snažíme o získání maximálního množství informací o tom, co uživatelé od nového systému očekávají a jaké činnosti bude systém primárně vykonávat. Mezi možné techniky zběru informací například patří:

- Studium dokumentů
- Rozhovory s budoucími uživateli
- Dotazníky
- Pozorování
- Nahrávání aktivit uživatele

Tyto požadavky se dají rozdělit na:

- **Funkční požadavky**. Tedy jaké jsou požadavky na funkčnost systému ze strany systému, případně se specifikuje i co by systém neměl umět. Základním způsobem pro popis funkčních požadavků je graf způsobů užití.
- **Nefunkční požadavky.** Jsou to požadavky na vlastnosti systému jako celku. Mohou se týkat výkonosti, spolehlivosti, bezpečnosti, kapacity systému, případně se zde mohou být organizační nároky, jako dodržované standardy a postupy práce. Případně zde mohou být zahrnuty požadavky týkající se platné legislativy.

Cíle

Hlavním cílem je pak vytvoření vize projektu. Tato vize by měla být definována z pohledu uživatele. Zároveň by ale měla obsahovat základní technický náhled na využité technologie, či komponenty architektury. V této vizi by měly být hlavně popsány požadavky na budoucí software. Tato fáze běžně proběhne jenom jednou. Ve složitějších případech ale může proběhnout vícekrát.

Zároveň by zde měla být zhodnocena míra rizik v oblastech:

- Rizika technického směru (technologií, kompatibilitu, komunikaci s ostatními systémy)
- Finanční rizika – náklady na vývoj a následnou údržbu

- Časová rizika – přičemž čím více budeme mít zdrojů, tím pravděpodobně kratší bude samostatný vývoj

Na konci této fáze je třeba posoudit, zda není vhodné skončit vývoj. Čím dříve ukončíme vývoj problematického projektu, tím jsou náklady na vývoj tohoto projektu nižší. K tomuto účelu se používá Lifecycle Objective Milestone (LOM) jeho základní kritéria jsou:

- Shoda zúčastněných stran na rozsahu, nákladech a časovém harmonogramu projektu
- Shoda na tom popsání správných požadavků na systém a chápání těchto požadavků je shodné
- Shoda správnosti finančních a časových odhadů, prioritách, rizicích a odpovídajícím vývojovém procesu
- Shoda v odhadu všech rizik a strategiích jejich snižování

4.3. Rozpracování (Elaboration)

Ve fázi zpracování se projekt začíná formovat. Cílem této fáze je popsat architekturu systému, abychom na jejím základě mohli v následující fázi navrhnout většinu funkcionalit. Tato architektura se vyvíjí na základě poznatků z předchozí fáze. Řeší se zde hlavní problémy týkající se návrhu architektury vznikajícího software. Funkčnost systému může být pouze přibližná. Většina kritických funkcionalit by již ale měla být do systému integrována. Řešitelský tým by si měl v průběhu této fáze klást otázky týkající se stability návrhu.

V této fázi se provádí analýza problému a architektura projektu získává základní podobu. Vhodným návrhem architektury se snažíme minimalizovat rizika spojená se vznikem software. Jde především o rizika:

- Finanční a časová
- Architektonická
- Technická a procesní
- Zaměření vyvíjeného software

Pomocí několika iterací v této části projektu můžeme snížit míru výše uvedených rizik. Hlavně pak správnost zaměření vývoje. Pokud tvoříme systém s využitím podobných technologií, jaké jsme již tvořili, existuje logicky menší množství potencionálních rizik. Můžeme si proto i dovolit naplnit cíle této fáze pouze v jedné iteraci. Naopak pokud technologii neznáme nebo je projekt z nějakého důvodu složitější (např. vyšší bezpečnostní požadavky). Pracujeme ve dvou, či více iteracích. V případě zásadnější změny architektury v průběhu této fáze je rozumné přidat další iteraci, která ověří dostatečnou funkčnost a stabilitu pozměněné architektury. Podle zásady, čím později budu zásadně měnit základy celé stavby, tím nákladnější bude celá oprava.

Pro ověření správného zaměření vyvíjeného software může být přínosné vytvořit beta

verzi uživatelského rozhraní, na kterém se otestují nejdůležitější funkcionality systému.

Na konci této fáze musí být architektura ve stabilním tvaru. Pro její ověření je třeba spustitelná architektura, na které ověříme kritické funkcionality systému. Tyto kritické funkcionality je také vhodné vymodelovat třeba například pomocí sekvenčního diagramu v UML. Snažíme se nalézt potencionální problémy a rizika. Následně bychom měli seskupit vybrané třídy do balíčků s ohledem na pravidla viditelnosti a budoucí konfiguraci produktu. Měli bychom si vytvořit představu, jak bude nakládáno s daty v databázi.

Pravidelně bychom pak měli provádět integraci komponent. Tedy určovat v jakém pořadí a jak budou vybrané komponenty integrovány.

Na závěr velmi důležitou částí je testování kritických scénářů. Testování kritických scénářů může být ukazatelem našeho pokroku ve fázi rozpracování. Mimo jiné je vhodné v této fázi testovat v kritických scénářích takové aspekty jako je velká zátěž systému, dostatečná výkonost architektury, či spolupráci s externími systémy.

Na konci této fáze dochází opět k otestování pomocí Lifecycle Architecture milestone (LCA) základními evaluačními kritérii LCA jsou:

- Jsou vize výroby a požadavky na výrobek stabilní?
- Je architektura stabilní?
- Jsou otestovány klíčové postupy a přístupy? A tím je ukázána jejich použitelnost.
- Pomocí testování spustitelného prototypu byly popsány hlavní rizikové části projektu. Byly tyto části následně vyřešeny
- Jsou připraveny iterační plány pro následující fázi v takové podobě, že podle nich půjde postupovat?
- Jsou tyto plány podpořeny důvěryhodnými odhady?
- Všichni zúčastnění se shodují v tom, že současná vize může být naplněna, pokud bude současný plán dotažen v kontextu současné architektury?
- Jsou akceptovatelné současné náklady oproti plánovaným?

Projekt může být přerušený nebo výrazně přehodnocen, pokud nedosáhne tohoto milníku.

4.4. Konstrukce (Construction)

Hlavním cílem této fáze je vybudovat reálně fungující softwarový systém a minimalizace nákladů na jeho tvorbu.

V této fázi se hlavní pozornost soustředí na vývoj komponent a dalších prvků systému. Toto je fáze, kdy dochází k větší části kódování, díky čemuž je lidsky (vyžaduje hodně programátorů a testerů) i časově náročná. Dochází zde k návrhu zbylých funkcionalit tedy zbytku hlavních (požadovaných zákazníkem) a většiny ostatních. Pokud jsou zde

zásahy do architektury systému, pravděpodobně byla špatně provedena předchozí fáze.

Hlavními předpoklady pro úspěch této fáze je: kompaktnost architektury, paralelní vývoj jednotlivých týmů, management konfigurací a změn a automatizované testování.

U větších projektů je možné projít několik iterací (běžně 2 – 4). Většinou v této fázi projektu bývá nejvíce iterací. Plánování iterací odpovídá počtu a typu ještě neimplementovaných funkcionalit. V každé iteraci se řeší individuální segment projektu. V ideálním případě je z předchozích fází vytvořena architektura, která se dále rozšiřuje, využívá, či znovu využívá. Probíhá integrace a testování systému. Organizačně je ideální jeden tým zodpovědný za architekturu a několik dalších týmů, které paralelně pracují na vývoji subsystému. Tyto „paralelní týmy“ hlavně komunikují s týmem zodpovědným za architekturu software.

Díky iteračnímu vývoji vzniká velké množství soborů, jejich verzí, které jsou následně testovány a integrovány. Z tohoto důvodu musí být zaveden management konfigurací a změn. Který jednotlivé změny a konfigurace zaznamenává. Pokud takový systém funguje, mohou se vývojáři věnovat vývoji, což jim ušetří čas, tedy zvýší efektivitu jejich práce.

Na konci této fáze musí být hotova beta verze programu včetně nápovědy v aplikaci, instalačních instrukcí, uživatelských manuálů a tutoriálů. Tak aby vybraní budoucí uživatelé mohli vyzkoušet software a dát nám relevantní zpětnou vazbu.

Na konci této fáze bychom si měli položit několik otázek podle milníku IOC (Initial Operational Capability)

- Je beta verze vhodná k uvolnění prvním uživatelů (testerům)?
- Jsou uživatelé připraveni a schopni používat naši beta verzi?
- Jsou akceptovatelné současné náklady oproti plánovaným?

4.5. Nasazení (Transition)

- Cílem této fáze je přesun systému z jeho vývoje do reálného využití a vychytání jeho posledních problémů (nejčastěji z pohledu výkonosti, uživatelské přívětivosti a funkcionality). Činnosti této fáze zahrnují:
- školení koncových uživatelů a správců
- testování systému, aby se ověřilo naplnění očekávání s koncových uživatelů.
- Příprava marketingových materiálů
- Příprava prostředí a dat, například migrace dat ze starého do nového systému
- Produkt je dále kontrolován v kontextu kvalitativních rámců vytvořených na začátku projektu.
- Vnitřní zhodnocení projektu, ponaučení se z chyb a problémů vzniklých při práci na projektu

Anglická wikipedie ze dne 24. 6. 18 dále uvádí: „Systém také prochází evaluační fází, každý vývojář, který nevytváří potřebnou práci, je nahrazen nebo odstraněn.“

I tato fáze by měl být zakončena milníkem. Tentokrát PRM (Product Release Milestone) Kritéria tohoto milníku jsou:

- Jsou uživatelé spokojeni
- Jsou akceptovatelné konečné náklady oproti plánovaným? Pokud co by se mělo změnit, aby se tomuto problému předešlo?

5. Business Process Model and Notation

Primárním cílem bylo poskytnout grafický nástroj, který je jednoduše pochopitelný všemi firemními uživateli (od analytiků, přes vývojáře až po vlastníky podnikových procesů). Díky čemuž se výrazně zjednoduší komunikace mezi vývojářem a zákazníkem. Business Process Model and Notation (Dále již jen BPMN) se skládá z množiny jednoduchých grafických prvků, ze kterých se dá vymodelovat firemní procesní model. BPMN je vyvíjen od roku 2005. Někdy v roce 2011 nastupuje verze 2.0. Procesní diagram je založen na vývojovém diagramu, který je velmi podobná diagramu aktivit z UML, o kterém budeme také mluvit.

Tokové objekty (Flow Objects)

Bezprostředně souvisí s tokem informací v procesu. Jsou to nejdůležitější grafické prvky. Spadají do nich tři skupiny prvků. Události, aktivity a brány.

Událost (Event)

Značíme je kolečkem, do kterého může být zobrazena různá ikona. Značí děj, který přímo ovlivňuje tok procesu. Události dále dělíme na **počáteční** (Start event), které iniciují vznik procesu a značí se kruhem s jednoduchým okrajem, někdy také zelenou barvou.

Dále máme **konečné události** (End event), které značí ukončení procesu, anebo výsledek aktivity. Je znázorněn tučným okrajem kolečka anebo tučnou ikonou uvnitř kolečka, občas také červenou barvou. V některých zdrojích se dále uvádí střední událost, která se pak značí dvojitým kruhem

Aktivita (Activity)

Značíme ji obdélník s kulatými rohy. Tento prvek znázorňuje činnost či práci. Aktivita může být buď **atomická** (tzv. Task), pak na ni pohlížíme jako na dále nedělitelný celek. Který se vždy musí vykonat celý.

Případně v sobě může aktivita obsahovat samostatný proces, pak ji nazýváme Subproces. Tento druh aktivity využijeme u procesů, u kterých nechceme, aby byly v dané úrovni znázorněny. Aktivitu Subproces značíme malým plus dole v zaokrouhleném obdélníku.

Brána (Gateway)

Značíme je čtvercem či kosočtvercem, stojícím na špičce. Označují větvení či souběh toků procesu, např. rozhodování či paralelní zpracování.

Spojovací objekty (Connecting Objects)

slouží ke spojení tokových objektů, či s artefakty. Díky spojení vzniká struktura (kostra) podnikového procesu. Spojovací objekty se dělí na 3 základní typy:

- **Sekvenční tok** (Sequence Flow) - plná čára s plnou šipkou. Určuje pořadí, v jakém

se mají jednotlivé řásti grafu vykonat

- **Tok zpráv** (Message Flow) - přerušovaná čára s prázdnou šipkou. Zobrazuje tok zpráv mezi jednotlivými účastníky procesu
- **Asociace** (Association) – tečkovaná čára. Spojuje objekt s nějakou dodatečnou informací. Využívá se také pro zobrazení vstupů a výstupů.

Plavecké dráhy (Swimlanes)

Zobrazuje účastníky procesu, nebo uspořádání činnosti v procesu typicky podle rolí. Používají se dva typy plaveckých drah.

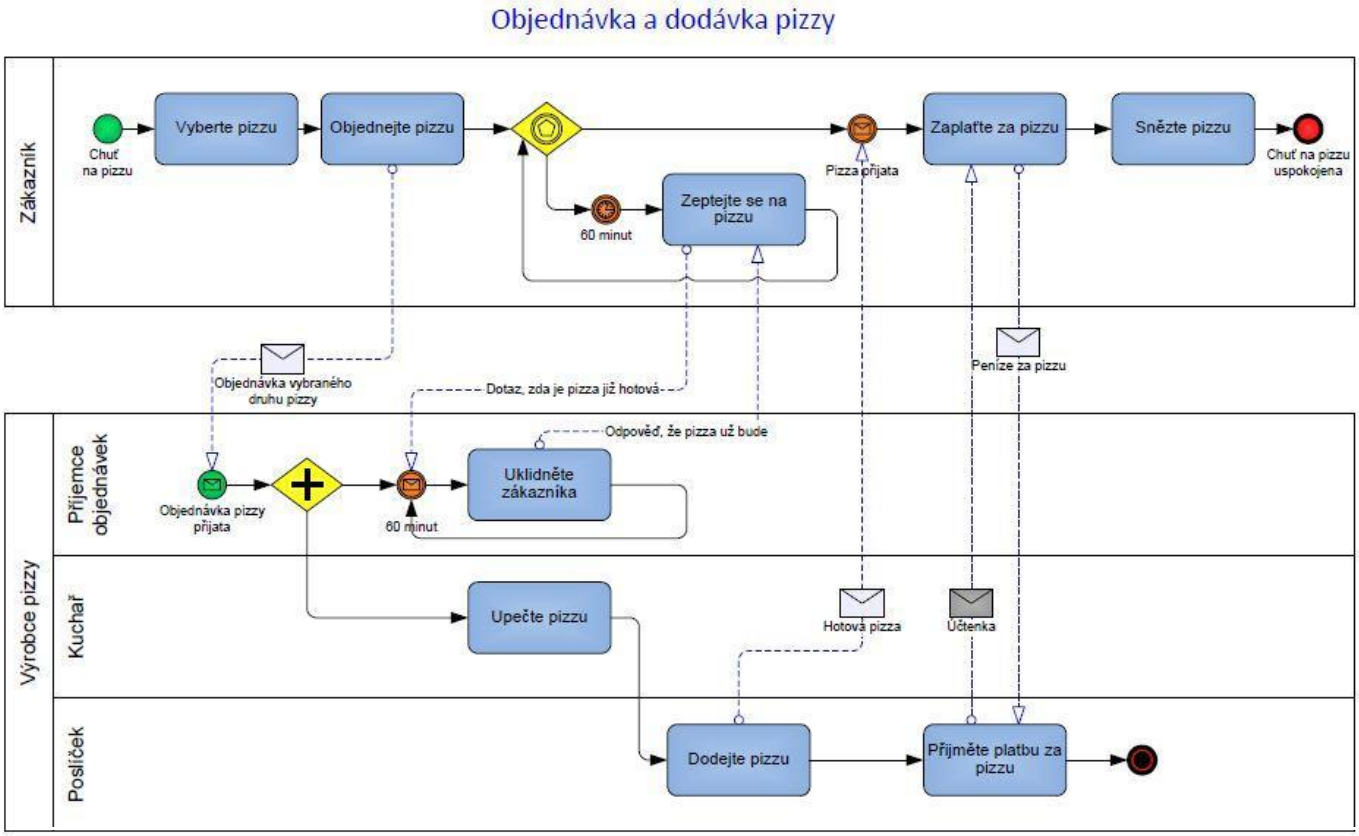
- **Bazén** (Pool) - Zobrazuje účastníky procesu, případně odděluje různé části organizace. Může mít více drah. V jednom bazénu se nachází jeden samostatný proces.
- **Dráha** (Lane) – pod částí bazénu. Používá se pro organizaci a kategorizaci aktivit

Artefakty (Artifacts)

Nějaké další upřesňující informace pro proces. Bez vlivu tok.

- **Datový objekt** (Data Object) – značíme obdélníkem se zahnutým rohem (list papíru). Značí data, se kterými pracují aktivity. Případně jsou nezbytná pro vykonání činnosti.
- **Seskupení** (Group) – značíme obdélníkem kresleným přerušovanou čárou. Seskupují různé objekty.
- **Poznámka** (Annotation) – Poznámky celému diagramu dodávají srozumitelnost a přehlednost

Jako vzor přikládáme model objednávky a dodávky pizzy.



6. UML

Mnohé softwarové systémy jsou velmi složité. Jejich vývoj tak klade velké požadavky na vývojářovu představivost. V momentě kdy na projektu pracuje více lidí, je třeba budoucí softwarový systém nějak exaktně popsat – modelovat. Existuje velká spousta přístupů k modelování. Každý z nich má své klady i zápory. Pro spolupráci více vývojářů je ale důležité, aby všichni znali, používali a hlavně stejně chápali modelovací systém a terminologii. Z tohoto důvodu se rozšířil UML (Unified Modeling Language, unifikovaný modelovací jazyk). Standard UML definuje standardizační skupina Object Management Group (OMG).

UML je jednotný jazyk pro vytváření určený k diagramů. UML jazyk umožňuje specifikaci (struktura a model), vizualizaci (diagramy), konstrukci (Software development methods) a dokumentaci artefaktů softwarového systému. Existuje velká řada diagramů. Každý z těchto diagramů se používá za rozdílným cílem a v rozdílných částech projektu. UML celkem poskytuje 13 druhů diagramů. V tomto textu budeme diagramy dělit na Strukturální diagramy a Diagramy tříd.

Strukturární diagramy

(Structure diagrams) - popisující strukturu projektu. Často se používání při popisu architektury vyvíjeného software. Ukážeme si několik vybraných diagramů spadajících do této kategorie

Diagram tříd (class diagram)

Diagram ukazuje členění tříd v projektu, vztahy mezi nimi, operace, nebo metody a jejich metody. Tyto části se často zapisují do obdélní vertikálně rozděleného na 4 části. Tento diagram je důležitým stavebním kamenem v objektově orientovaném modelování. Může se také sloužit k datovému modelování. Diagram zobrazuje statickou strukturu modelovaného systému. A je specificky závislý na konkrétním programovacím jazyku. Cílem této aktivity je ukázat jakým způsobem bude produkt realizován v implementační fázi.

Níže příklad diagramu tříd.

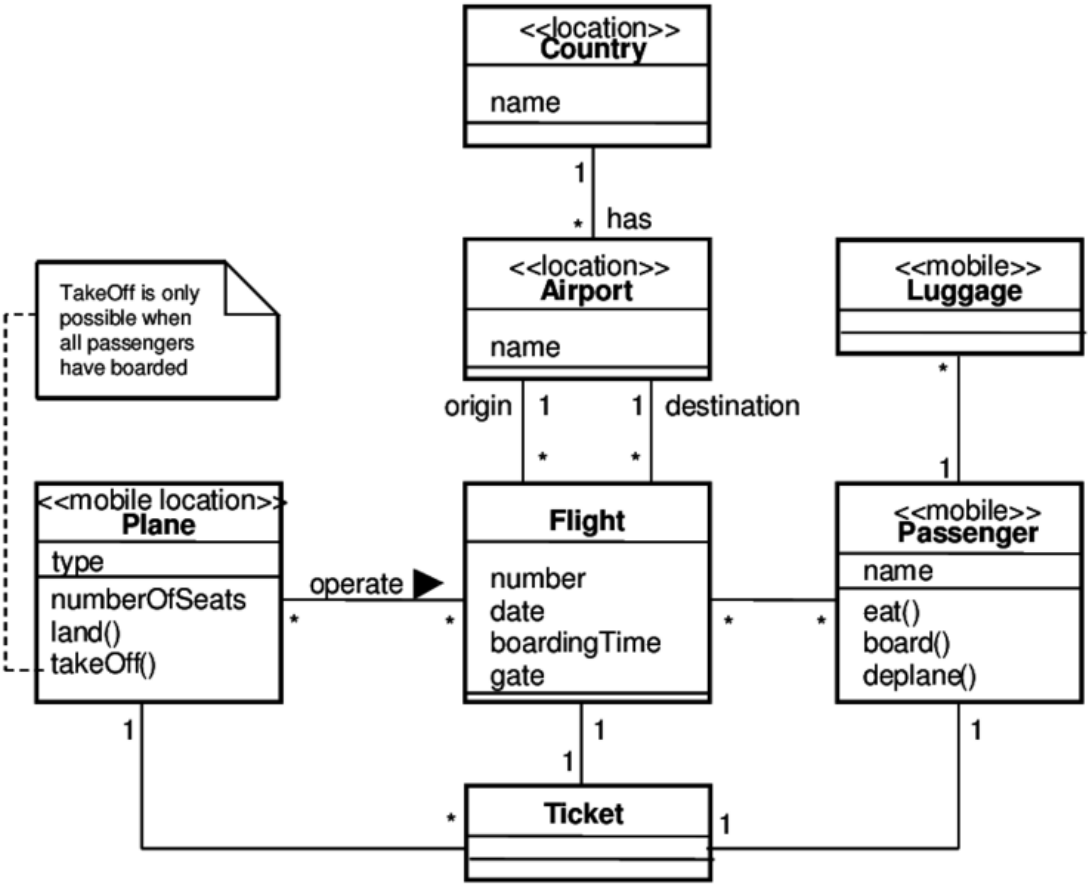


Figure 3 Zdroj: Andrade et al, Recent Trends in Algebraic Develeopment Techniques—16th International Workshop, WADT 2002, Frauenchiemsee, Germany. Vol. 2755 of LNCS.

Diagram komponent

Diagram pracuje s komponentami využitými v systému a znázorňuje, jak jsou komponenty propojeny k vytvoření větších komponent nebo softwarových systémů. Jsou používány k popisu struktury různě složitých systémů. Tento diagram se využívá hlavně při tvorbě software pomocí komponent. Tento diagram má vyšší úroveň abstrakce, než diagram tříd. Většinou komponenta v sobě implementuje jednu anebo více tříd.

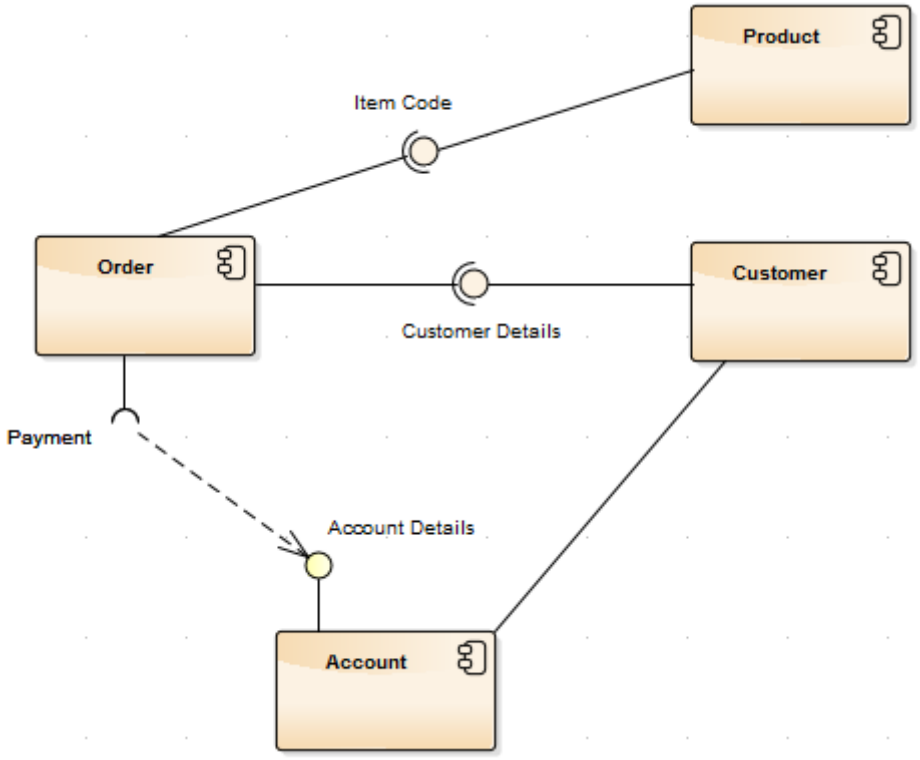
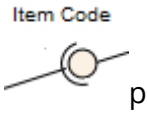


Figure 4 Zdroj:

http://sparxsystems.com/enterprise_architect_user_guide/13.0/model_domains/componentdiagram.html

V našem obrázku máme celkem 4 komponenty.



Montážní konektory (Assembly connectors) propojují rozhraní objednávky s produktem a zákazníkem. V našem případě komponenta vyžaduje informace ID produktu, a detaily zákazníka.

Prostá čára mezi komponentami Zákazník a učet zákazníka znázorňuje vztah mezi komponentami.

Objektový diagram

Objektový diagram znázorňuje objekty (instance tříd) a vztahy mezi nimi (links – instance asociací) právě v jednom okamžiku. Tento diagram vypadá podobně jako diagram tříd a vlastně může být jeho speciálním případem. Hlavně se využívá, pokud chceme ukázat příklady datových struktur v jeden konkrétní moment.

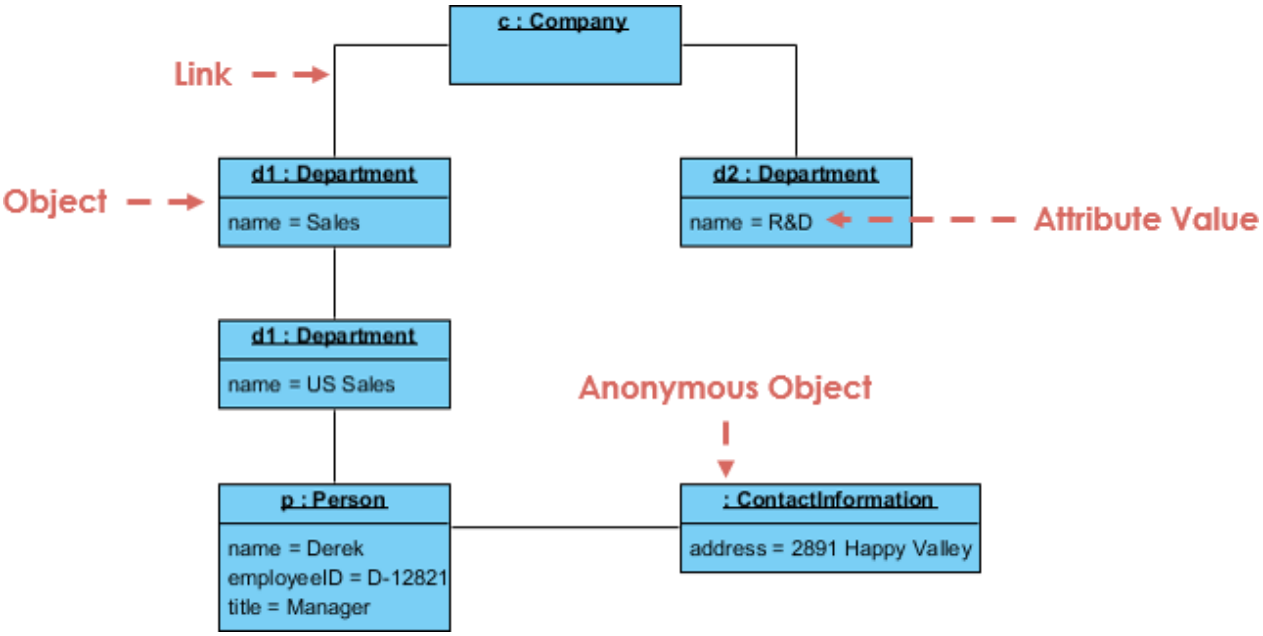


Figure 5 Zdroj: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-object-diagram/

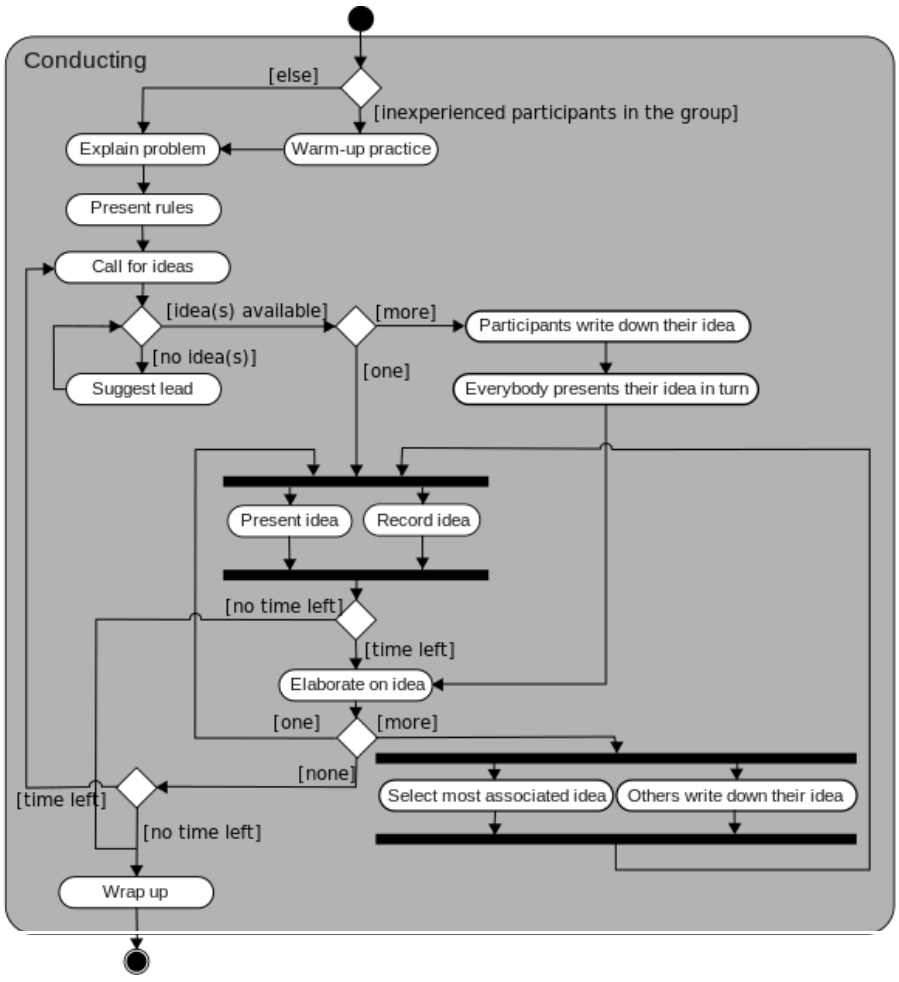
7. Diagramy chování (Behavior diagrams)

Popisující jednotlivé procesy pomocí aktivit reprezentujících jeho stavy a přechody mezi nimi. Vzhledem k tomu, že diagramy aktivit popisují chování systému, používají se k popisu funkčnosti softwarových systémů.

Diagram aktivit (Activity diagram)

Diagram je určen k modelování výpočetních či organizačních procesů (tj. Pracovních postupů) lze jej použít i k modelování datových toků. Diagram modeluje procesy pomocí aktivit reprezentujících jeho stavy a přechody mezi jednotlivými stavy. Díky tomu je jedním z diagramů, které umožňují zachytit chování systému.

- zaoblené obdélníky představují akce
- kosočtverce představují rozhodnutí
- čáry se šipkami spojují jednotlivé akce
- černý kruh představuje počátek (počáteční uzel)
- zakroužkované černý kruh představuje konec (konečná uzel)



Sekvenční diagramy

Diagram sekvence zobrazuje interakce objektů, které se podílejí na konkrétní funkcionalitě systému. Krom objektů a tříd zobrazuje posloupnost zpráv nutných k dosažení dané funkcionality, mezi jednotlivými objekty a to v podobě, která řeší hlavně jejich časovou návaznost. Tyto diagramy se často objevují v souvislosti s realizací případů použití v logickém pohledu na vývojový systém. Cílem této aktivity je ukázat jakým způsobem bude produkt realizován v implementační fázi.

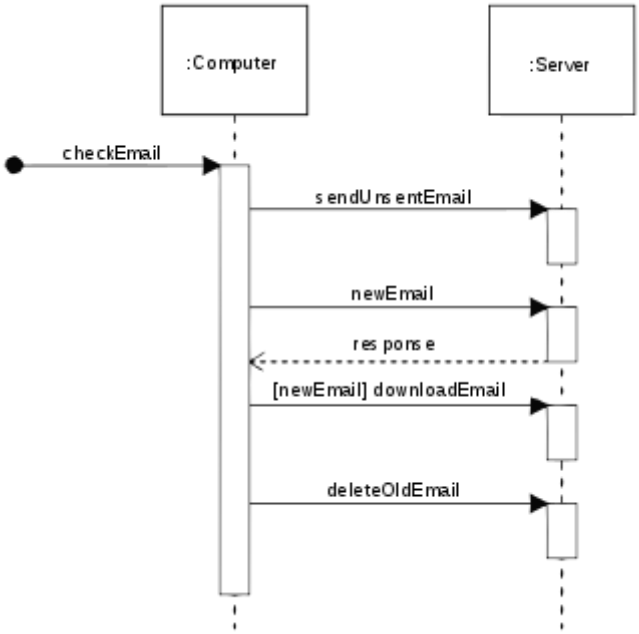


Figure 7 Zdroj: https://en.wikipedia.org/wiki/Sequence_diagram#/media/File:CheckEmail.svg

Časová osa v diagram je svislá (čas běží zhora dolů). Objekty jsou rozmístěny vodorovně. V našem obrázku máme 2 objekty (Server a počítač) čárkovaná svislá čára také nazývá čára života. V našem případě objekty existovali a budou existovat před i po diagramu. Obdélníky na čáře života znázorňují aktivitu. Vodorovné čáry se šipkou znázorňují zprávy. Šipka od koho komu zpráva půjde. Plná čára značí povinou zprávu. Čárkovaná nepovinou.

Diagram případů užití (Use case diagram)

Nejčastěji znázorňuje vztah mezi zákazníkem a systémem. Kdy znázorňuje různé případy využití systému zákazníkem. Což je často slovně popsáno pomocí scénářů. Pro převod scénářů do diagramů případů užití se využívají v čase realizace. Jde o algoritmus přepisu bodů scénáře do diagramu.

Vstupem pro vznik tohoto diagramu může být například BPMN. Výsledkem je seznam aktivit, které může například místo lidí vykonávat právě vznikající software. Postavičky v diagramu se nazývají aktér (actor). Aktéři jsou tedy propojeni s těmi případy užití, které se jich týkají. Jednoduchá čára se nazývá asosiace. Horizontální směr dále naznačuje strukturu aktérů. Spodní aktér dědí vlastnosti aktérů nad ním. Tento typ vazby je nazýván generalizace. Vazba include se realizuje vždy, když je realizován případ užití, ze

kterého tato vazba vychází.

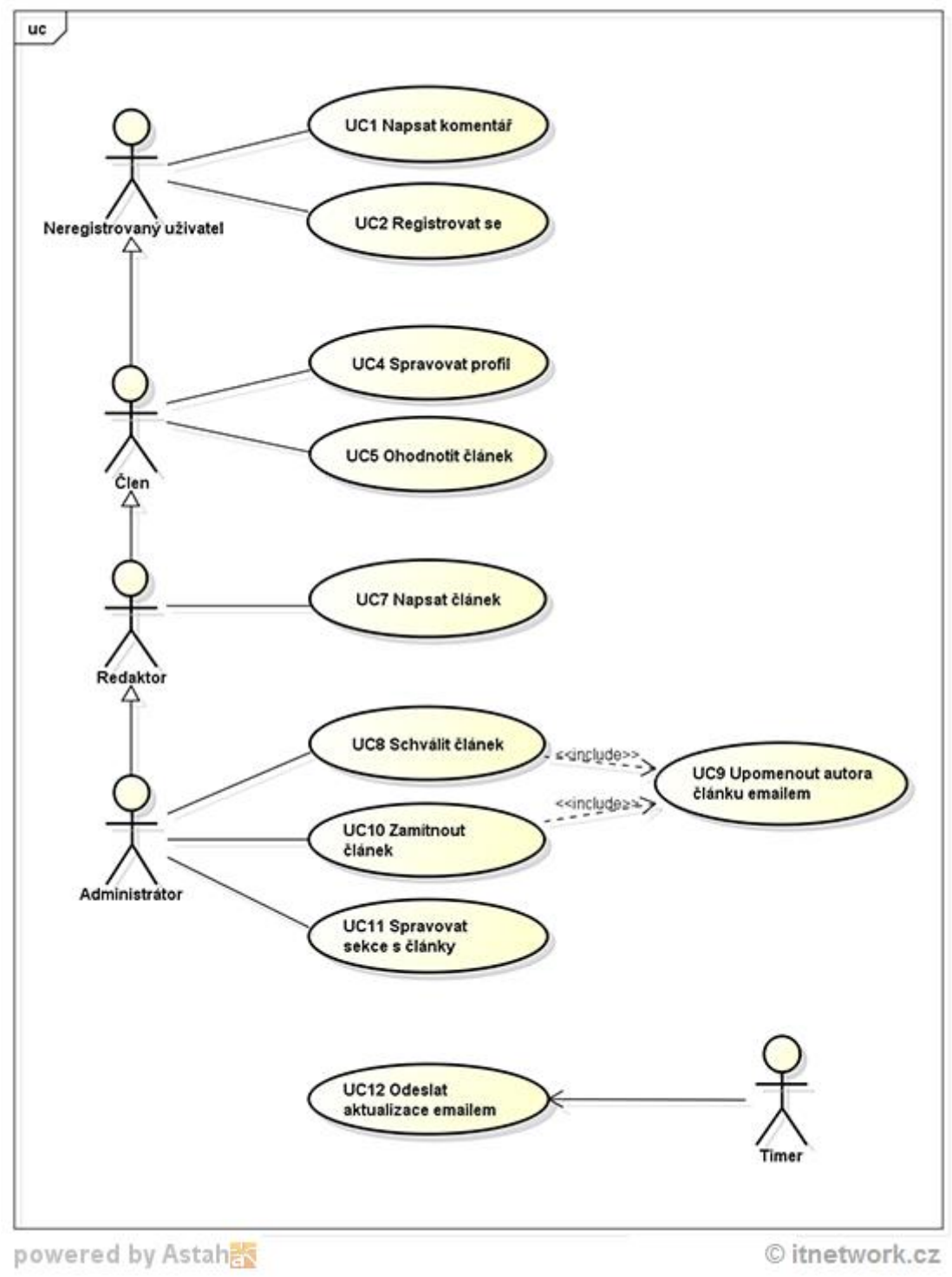


Figure 8 Zdroj: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

7.1. Testování a nasazení vytvářeného software

Existuje velké množství testů, přičemž každý test je většinou zaměřen na jinou oblast testování. Jednotlivé testy lze rozdělit na funkční a nefunkční. Funkční testy jsou zaměřeny na přímé požadavky uživatele. Cíle těchto testů můžeme například definovat z pohledu standardu ISO/IEC 12207 nebo z výsledků zjištěných pomocí FURPS+.K příslušnému požadavku následně vytvoříme test.

Nefunkční testy netestují přímo funkčnost systému, testují jeho vlastnosti, které uživatel běžně automaticky očekává. Jde například o:

- Bezpečnostní testy - testují bezpečnost systému
- Zátěžové testy testující chování systému při zátěži.
- Testy použitelnosti – budou použitelné za různých omezujících podmínek. Třeba rychlost připojení k internetu
- Testy dokumentace – u uživatelské dokumentace testujeme srozumitelnost a konzistetntnost. U vývojářské pak množství pokrytí

Softwarové systémy se také testují z pohledu verifikace a validace.

- Pokud **Verifikujeme,** snažíme se zodpovědět otázku. Zda-li je produkt vytvářen a nebo již vytvořen správně po technické stránce.
- Pokud **Validujeme,** odpovídáme na otázku: Slouží software k tomu, k čemu by měl sloužit?

Například pokud zákazník chce systém e-shop Námi vytvořený systém má mnoho zábavných funkcí bohužel možnost koupit je schována někde hluboko, takže většina zákazníku ji nenajde, tudíž nenakoupí.

Někdy se také mluví o Black box a White box testování.

- U **Black box** testování přestupujeme k vyvíjenému softwarovému systému jako k black boxu. Neřešíme tedy co je uvnitř. Kontrolujeme pouze funkcionalitu z pohledu uživatele. Nejčastěji porovnáváme, zda na zadané vstupy vychází odpovídající výstupy. Snažíme se při tom najít i nezvyklé, či neočekávané vstupy, na které by systém nemusel reagovat korektně
- U **White box** testování známe strukturu programu. Ověřujeme zda, že jednotlivé větve programu fungují bez problémů. Snažíme se otestovat každou část programu.

Pro samotné nasazení softwarového systému u uživatele zmiňuje Vondrák (2002) následující aktivity:

- vytvoření výsledného produktu či jeho verzí
- kompletace softwarového systému
- distribuce softwarového systému
- instalace softwarového systému u uživatele
- poskytnutí asistenční služby uživatelům
- plánování a řízení beta testování
- migrace již existující dat a softwarových produktů

8. Měření kvality softwarových systémů

Pokud na konci procesu vývoje je spokojený uživatel/zákazník, můžeme předpokládat, že jsme vytvořili kvalitní software. Tento ukazatel je ale velmi subjektivní. Navíc každý uživatel/zákazník má jiné a jinak náročné očekávání. Proto je třeba najít nějaké objektivnější kritéria hodnocení kvality a přínosů softwarových systémů.

Tato kritéria se nazývají metrikami. (Je to poněkud nešťastný název, protože v matematice má jiný význam – zobecnění vzdálenosti)

Žáček (2017) definuje metriku takto:

Metriku můžeme definovat následovně: „Metrika je přesně vymezený finanční či nefinanční ukazatel nebo hodnotící kritérium, které je používáno k hodnocení úrovně efektivnosti konkrétní oblasti řízení podnikového výkonu a jeho efektivní podpory prostředky IS/ICT. Skupinu metrik sdružených za určitým cílem (tzn. vztahujících se ke konkrétní oblasti, procesu či projektu) nazýváme „portfolio metrik“

Metriky se většinou rozdělují na tvrdé a měkké.

- Tvrdé metriky – jsou objektivně měřitelné. Mezi příklady může být průměrná doba odezvy, počet chyb za časovou jednotku či délka záruky
- Měkké metriky – Jsou pak více subjektivní, těžko objektivně měřitelné. Mohou se vyhodnocovat například ve stupnicích typicky 0 – 100, v pořadí porovnávaných produktů případně i slovní hodnocení produktu.

Dále pak existují celé systémy pro řízení kvality vývoje softwarových systémů. Zákazník je pak většinou velmi rád, když jeho produkt má certifikát dokládající kvalitu. Mnohdy si za něj i rádi připlatí. Tyto systémy nám mohou pomoci k minimalizaci chyb a zvýšení efektivity vývojového procesu. My zde krátce pohovoříme o některých z nich.

8.1. CMMI

Capability Maturity Model (Integration). Tento standard je rozšířen hojně v USA a Japonsku. Je určen pro vývojové týmy. Je to poměrně podrobný model, díky čemuž může mít návodný charakter ke zlepšení úrovně vývojového týmu. Je to soubor pravidel a doporučení, která by měly vývojové týmy dodržovat pro efektivní vývoj a plánování nových produktů. Zaměřuje se na organizaci, plánování a sledování vývojových procesů. Jeho specifikem je rozdělení vývojových týmů do pěti úrovní schopností a zralostí. Jednotlivé týmy se pak mohou v tomto žebříčku posouvat. Posouzení úrovně schopností posuzuje vyškolený posuzovatel přesně daným postupem.

Stupně zralosti jsou (podle wikipedie)

- Počáteční (Initial): Týmy na této úrovni definované procesy nevykonávají nebo pouze částečně

- Řízená (Managed): Je stanoveno řízení projektů a činnosti jsou plánovány
- Definovaná (Defined): Postupy jsou definovány, dokumentovány a řízeny
- Kvantitativně řízená (Quantitatively Managed): Produkty i procesy jsou řízené kvantitativně
- Optimalizující (Optimizing): Tým soustavně optimalizuje své činnosti

Úrovně schopnosti

- Neúplné (Incomplete): Některé činnosti nejsou vykonávány
- Vykonávané (Performed): Činnosti jsou vykonávány
- Řízené (Managed): Je řízeno, jak jsou činnosti vykonávány
- Definované (Defined): Je definováno, jako jsou činnosti vykonávány

8.2. ISO 9000:2001

Je psán obecněji než CMMI. Definuje systém managementu kvality. Tato norma umožňuje prokázat daným organizacím schopnost výroby či distribuci produktů. Je tedy psána velmi obecně. Proces vývoje je dosti specifický. Z tohoto důvodu vznikla například Anglicko – Švédská interpretace TickIT a nebo ISO/IEC 90003

8.3. Six Sigma

Je souhrn postupů řízení. Jejíž cílem je pochopení a následné neustálé průběžné jejich zlepšování. Poznání zákazníka a jeho očekávání. Pomocí porozumění potřeb zákazníků, analýzy procesů, standardizace metod měření a jejich analýzy pomocí statistických metod.

Mezi základní filozofické předpoklady této metodologie patří:

- Neustálé úsilí o dosažení stabilních a předvídatelných výsledků procesu má zásadní význam pro obchodní úspěch.
- Výrobní a obchodní procesy mají vlastnosti, které lze definovat, měřit, analyzovat, zdokonalovat a kontrolovat.
- Dosažení trvalého zlepšování kvality vyžaduje angažovanost celé organizace, od nejvyšší úrovně řízení.

Mezi základní techniky patří například cyklus DMAIC – určeného k pochopení a řízení procesů.

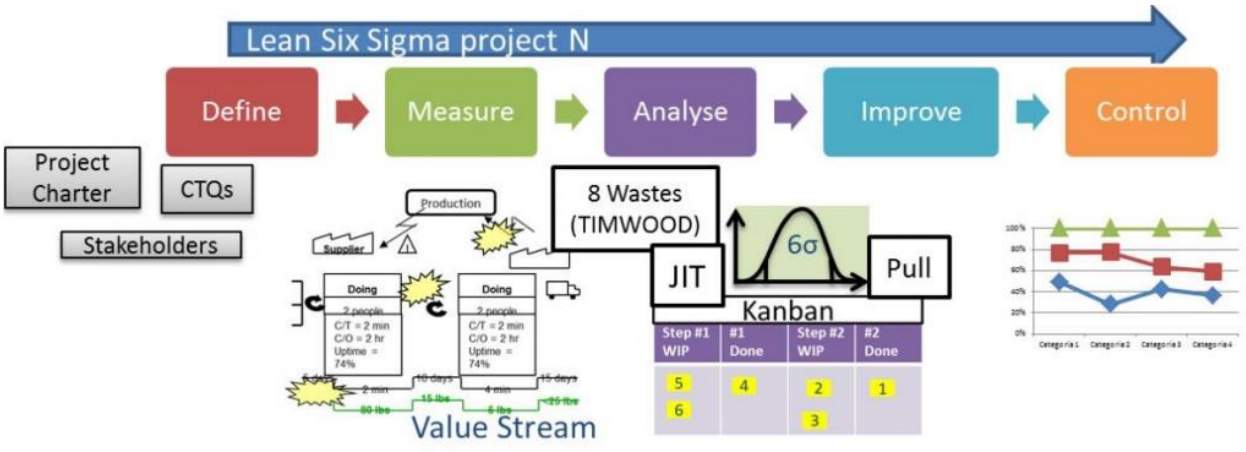


Figure 9 Zdroj: http://www1.osu.cz/~zacek/infs2/02.pdf

DMAIC obsahuje celkem 5 fází.

- **Define (definovat)** – popsání reálného procesu a ideálního procesu. Navrhuje se plán cesty od reálného k ideálnímu procesu
- **Measure (měřit)** – měření současného procesu. (Třeba průměrná doba čekání), tak abychom do rukou dostali „tvrdá data" snažící se popsat „objektivní realitu".
- **Analyze (analyzovat)** – Výsledky měření jsou statisticky analyzovány. Abychom mohli vytvořit fungující model konkrétního projektu. Případně zjistit, které části procesu je vhodné zlepšit.
- **Improve (zlepšovat)** – Realizace myšlenek vzniklých při předchozí fázi. Běžné cíle jsou zvýšení spokojenosti zákazníka, zvýšení efektivity, snížení nákladů.
- **Control (řídit)** – podařilo-li se změnu úspěšně zavést. Je třeba ji standardizovat. A kontrolovat, že přinesla nějaké zlepšení

Původně byla vytvořena ve společnosti Motorola. V současnosti je využívána například ve firmách Honeywell, HP, Texas Instruments, NASA a mnoha dalších. Je založena na aplikovaných statistických postupech, doplněných o kvalitativní nástroje.

9. Údržba a provoz softwarových systémů

Do této chvíle jsme se zabývali vývoje software. Vývojem ale životní cyklus software teprve začíná. Velmi důležitou částí je jeho samotný provoz. Pokud bychom vytvořili báječný software, který po měsíci provozu začne pravidelně kolabovat kvůli jeho špatné údržbě, zákazník by jistě měl pocit, že vývoj software nebyl dobře odveden. Je tedy důležité se náležitě věnovat také údržbě a provozu software.

Způsobům údržby a provozu obecně informačních technologií se věnuje ITIL. Je to souhrn konceptů a postupů, které umožňují obecně lepší využití IT služeb. IT službou rozumíme souhrn IT systémů, jejichž cílem je podpora podnikových procesů (tedy aby podnik dělal to co má).

ITIL je fyzicky sada knižních publikací, které obsahují souhrn nejlepších zkušeností z oboru řízení služeb informačních technologií. Je tedy frameworkem sloužícím ke správě IT služeb, který nám říká, kdy a co máme dělat. Myšlenka ITIL je celkem jednoduchá, proč navrhovat a vymýšlet celý proces znovu od začátku, když už ho spousta jiných firem má zavedený a průběžně ho i vylepšuje.

ITIL je zástupce proaktivního přístupu. Který se snaží problémy řešit dopředu, ne až zpětně. Pokud již nějaké problémy nastanou, snaží se je vyhledávat aktivní detekcí. K čemuž se snaží nastavit odpovídající procesy. Celý proces je běžně řízen, monitorován, měřen, vyhodnocován a neustále vylepšován. Všechny tyto procesy by měly být navrženy za účelem přidané hodnoty zákazníkovy. V některých případech může být problém v terminologiích, kdy rozdílné společnosti používají rozdílné názvy. ITIL se proto snaží sjednotit terminologii. Tyto postupy jsou navrženy jako platformě neutrální.

Pravděpodobně hlavním kladem těchto přístupů je zefektivnění. Díky kterému například daří zkracovat výpadky IT systémů.

Z tohoto důvodu byly zavedeny dva základní procesy Service Support a Service delivery. V rámci těchto procesů je zaveden Service desk – zákazníkovo kontaktní místo. Toto místo shromažďuje požadavky zákazníků či uživatelů. Má evidenci IT procesů. Díky čemuž může na požadavky reagovat a řešit je. Plní roli základní podpory.

V případě problémů nastupuje proces Incident management, který se snaží minimalizovat pro zákazníka nepříjemné důsledky problémů IT systémů. Jedno z nejdůležitějších kritérií je zde rychlost vyřešení problému.

Incident management má dále podporu od **Problem Management** který zpravuje evidenci řešení problémů. Zároveň analyzuje nastalé problémy a jejich trendy, a pokud je to vhodné navrhuje strukturární změny IT systémů.

Jednotlivé změny eviduje **Change managememement**. **Release management** pak plánuje a řídí jednotlivé změny IT služeb (release)

ITIL verze 3 se skládá z 5ti plus jedna úvodní kniha (Žáček, 2017) je charakterizuje takto:

- Service Strategy – zabývá se sladěním byznysu a IT, strategií správy IT služeb,

plánováním.

- Service Design – řešení IT služeb, návrh procesů (tvorba a údržba IT architektury, postupů).
- Service Transition – předání IT služby do byznys prostředí.
- Service Operation – doručení a řídící aktivity procesu, správa aplikací, změn, provozu, metrik.
- Continual Service Improvement – hnací body zlepšení IT služeb, oprávněnost vylepšení, metody, praktiky, metriky.

10. Literatura

Eysenck, M. W., & Keane, M. T. (2008). Kognitivní psychologie. Praha: Academia.

Nolen-Hoeksema, S. (2012). Psychologie Atkinsonové a Hilgarda (Vyd. 3., přeprac.). Praha: Portál.

Sklenář, V. (2007). SOFTWAROVÉ INŽENÝRSTVÍ [Online]. Retrieved June 29, 2018, from https://phoenix.inf.upol.cz/esf/ucebni/syspro.pdf

Softwarové inženýrství [Online]. (2001-). In Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation. Retrieved from https://cs.wikipedia.org/wiki/Softwarov%C3%A9_in%C5%BEen%C3%BDrstv%C3%AD#Softwarov%C3%A1_krize

Plháková, A. (2004). Učebnice obecné psychologie. Praha: Academia.

Rational Unified Process [Online]. (2001-). In Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation. Retrieved from https://en.wikipedia.org/wiki/Rational_Unified_Process

Rychta, A. (2011). Softwarové inženýrství [Online]. Retrieved June 29, 2018, from http://www.ksi.mff.cuni.cz/~richta/NSWI026/NSWI026-1-Uvod.pdf

Říčan, J. (2016). Používané metakognitivní strategie žáků pátých tříd ve specifické doméně čtení (Disertační práce). Praha.

US Department of Justice (2003). INFORMATION RESOURCES MANAGEMENT Chapter 1. Introduction.

Vondrák, I. (2002). Úvod do softwarového inženýrství [Online]. Retrieved June 29, 2018, from http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf

WHITE, Stephen A. Business Process Modeling Notation [online]. [cit. 2018-06-26]. Dostupné z: https://is.muni.cz/el/1433/jaro2014/PV165/um/46771256/pr_06_bpmn.pdf

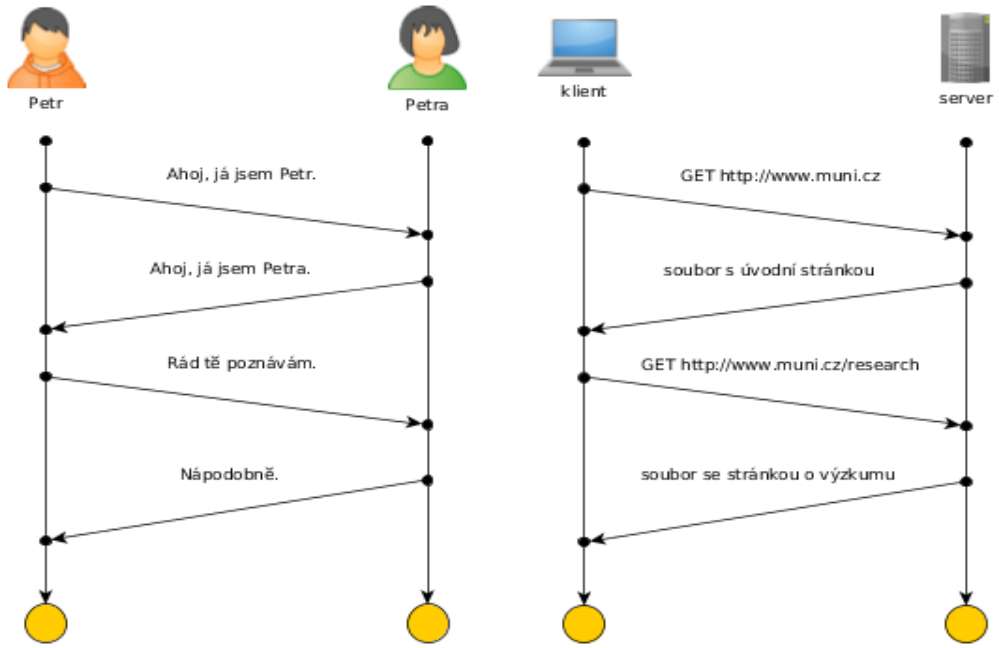
Žáček, J. (2017). SOFTWAROVÉ INŽENÝRSTVÍ [Online]. Retrieved June 29, 2018, from http://www1.osu.cz/~zacek/sweng/skripta_sweng.pdf

ZÁKLADY WEBOVÝCH APLIKACÍ

1. Komunikace, sítě, protokoly

1.1. Komunikační protokoly

V kontextu počítačových sítí se často setkáváme s komunikačními protokoly. Ty přesně definují způsob, jakým probíhá komunikace realizující konkrétní funkci a to na všech úrovních. Máme tak protokoly pro zasílání dat, navazování zabezpečených kanálů, vyhledání síťové adresy odpovídající doménovému jménu, doručení emailu atd. Protokol je známý oběma komunikujícím stranám a popisuje přesně jaký obsah, v jakém pořadí a s jakým časováním je předáván. Odklon od takto strukturované komunikace je možné interpretovat jako chybu.



Obrázek 1: Ilustrace komunikačního protokolu

1.2. TCP/IP

Základním principem prostupujícím architekturu počítačových sítí je rozdělení komunikace do vrstev podle abstrakce. Každá vrstva je zodpovědná za popis přenosu od úrovně aplikace až po komunikaci po fyzických spojích. Síťový model TCP/IP je základním kamenem všech dnešních sítí a i celého internetu a je pojmenován podle dvou hlavních protokolů zajišťujících směrování a transport dat mezi uzly. Protokol IP popisuje adresaci

uzlů, rozklad dat na pakety a jejich směrování uvnitř sítě.

Komunikace mezi stejnými vrstvami dvou různých systémů je řízena komunikačním protokolem za použití spojení vytvořeného sousední nižší vrstvou. Architektura umožňuje výměnu protokolů jedné vrstvy bez dopadu na ostatní. Architektura TCP/IP je členěna do čtyř vrstev (na rozdíl od referenčního modelu OSI se sedmi vrstvami):

- aplikační vrstva (*application layer*)
- transportní vrstva (*transport layer*)
- síťová vrstva (*internet layer*)
- vrstva síťového rozhraní (*network interface*)

Obrázek 2: Vrstvy síťového modelu TCP/IP

TCP je transportní protokol stavící „spojovanou" službu nad IP přenosem: zajišťuje kontrolu úspěšnosti přenosu a případné přeposílání chybějících či poškozených částí a volí optimální rychlost přenosu.

Rodina protokolů TCP/IP, kterou dnes používáme k realizaci naprosté většiny síťové komunikace, byla navržena na přelomu 70. a 80. let s cílem vytvořit robustnější model síťové komunikace, který by byl schopen se do určité míry vypořádat i s výpadky částí sítě. Základní (a v té době celkem revoluční) myšlenkou je *packet switching* (přepínání paketů): data nejsou posílána jako souvislý proud (stream), ale po samostatných blocích (paketech), a jednotlivé uzly sítě samy rozhodují, kudy budou pakety dále posílat.

V praxi je tato myšlenka realizována sadou protokolů, implementujících potřebné funkce. Protokoly obvykle rozdělujeme do několika úrovní (vrstev). Místo abstraktního ISO/OSI modelu, který pracuje se sedmi vrstvami, při výkladu TCP/IP většinou používáme zjednodušený pětivrstvý model (některé protokoly v TCP/IP modelu zastávají funkci více vrstev ISO/OSI modelu).

1.2.1. APLIKAČNÍ VRSTVA

Nejvýše je *aplikační vrstva*, tak označujeme data aplikačních protokolů jednotlivých síťových služeb. zahrnuje protokoly síťových aplikací: elektronické pošty, HTTP (webové stránky), DNS (doménová služba) a další .Takových protokolů existuje obrovské množství, z nejznámějších uveďme např. HTTP, SMTP, FTP, NTP. Z pohledu TCP/IP se jedná o data, která je třeba přenést k cílovému příjemci. O to se starají nižší vrstvy.

1.2.2. FYZICKÁ VRSTVA

Nejníže je v modelu vrstva *fyzická*. Na rozdíl od vyšších vrstev se nejedná o softwarovou vrstvu (protokol), tímto označením rozumíme konkrétní fyzické médium, které používáme k přenosu dat. Příkladem může být např. twisted pair (kroucená dvoulinka) kabeláž ve většině lokálních ethernetových sítí, koaxiální kabel, optické vlákno nebo telefonní linka. Médium ale nemusí být hmotné - např. v případě bezdrátových sítí v mikrovlnném pásmu (wi-fi, breezenet) nebo optických pojítek.

1.2.3. LINKOVÁ VRSTVA

Nejníže ze softwarových vrstev je *linková vrstva*. Jedná se o nejnižší komunikační protokol, sloužící k přenášení dat po fyzickém médiu. Tento protokol je většinou úzce svázán s konkrétní volbou média, ale tato korespondence nemusí být 1:1, např. ethernet bývá v praxi implementován nejen na twisted-pair kabeláži, ale můžeme se setkat s jeho implementacemi pomocí koaxiálního kabelu nebo naopak optických vláken. Jiným příkladem protokolu linkové vrstvy je PPP, protokol používaný k realizaci vytáčeného připojení (dial-up) nebo propojení počítačů přes sériovou linku. Podstatnou vlastností protokolů linkové vrstvy je skutečnost, že řeší pouze komunikaci mezi uzly, které jsou *přímo* spojeny (odtud i název).

1.2.4. SÍŤOVÁ VRSTVA

Globální adresaci a směrování má na starosti vrstva *síťová*, v praxi realizovaná téměř výhradně protokolem IP (vyskytujícím se ve dvou verzích, IPv4 a IPv6). Zatímco i u protokolů linkové vrstvy existují adresy a např. v případě ethernetových MAC adres jsou (nebo by aspoň měly být) dokonce globálně jednoznačné, nelze je použít ke směrování paketů, protože z takových adres nelze poznat, kde cíl hledat. Adresy protokolu IP (IP adresy) jsou ale přidělovány hierarchicky tak, že delegace jednotlivých rozsahů odpovídá topologii sítě. Z cílové IP adresy lze proto určit, kudy máme paket dále poslat, tedy alespoň následujícího prostředníka (hop) po cestě. Kromě této své základní funkce řeší protokol IP ještě některé další, např. fragmentaci (rozdělení příliš dlouhých paketů na několik kratších) nebo označení paketů podle typu provozu (ToS - type of

service).

1.2.5. TRANSPORTNÍ VRSTVA

Nejpoužívanějšími protokoly transportní vrstvy jsou dnes UDP a TCP. UDP (User Datagram Protocol) lze chápat jako minimalistický transportní protokol, zavádějící pouze pojem portu, který lze chápat jako adresu konkrétního procesu (přesněji socketu) v rámci cílového uzlu. UDP je ale stále bezstavový protokol (nelze tady mluvit v pravém slova smyslu o spojení), neřeší otázku ztracených paketů ani jejich pořadí. Přesto se často používá pro svou jednoduchost a nižší režii, a to zejména tam, kde tyto otázky řešit nepotřebujeme.

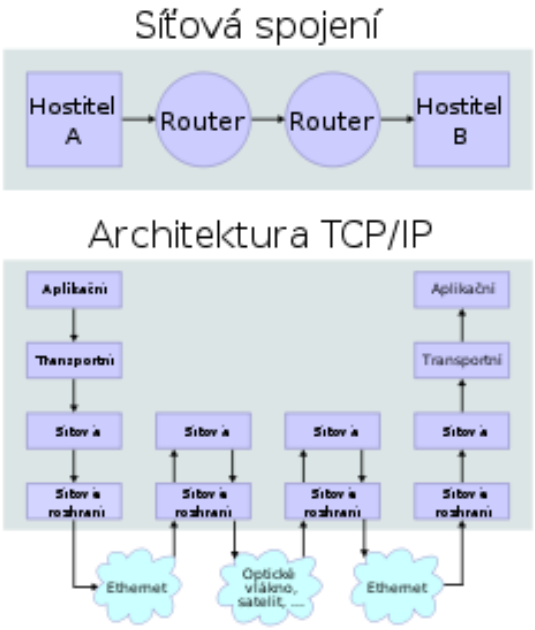
Opačný přístup je reprezentován protokolem TCP (Transmission Control Protocol). Ten naopak zavádí *spojení* mezi dvěma porty na koncových uzlech. Z pohledu klientské aplikace se takové spojení chová podobně jako roura pro komunikaci mezi dvěma procesy (ale na rozdíl od roury je TCP spojení obousměrné), je zaručeno, že posloupnost bytů (stream), kterou jedna strana odešle, dostane ve stejné podobě druhá strana. Protokol TCP se stará o detekci a opakované odeslání ztracených dat, stejně jako o přerovnání dat z paketů, které dojdou ve špatném pořadí. TCP tak poskytuje aplikační vrstvě poměrně vysokou míru komfortu, většina komunikace proto dnes používá jako transportní protokol TCP. Nevýhodou ale může být vyšší režie a relativně pomalá reakce na výpadky, proto se pro některé účely (např. většina DNS dotazů nebo VoIP) dává přednost UDP.

1.3. ICMP

Ze struktury vrstev se trochu vymyká protokol ICMP (Internet Control Message Protocol). Pakety (message) tohoto protokolu jsou přenášeny přímo prostřednictvím IP protokolu, ICMP ale nelze považovat za transportní protokol, protože neslouží k přenášení aplikačních dat. Tento protokol slouží k diagnostickým a servisním účelům. Příkladem aplikací ICMP jsou zprávy o nedoručitelnosti paketu (destination unreachable), pakety generované příkazem echo (ICMP echo a echo reply) nebo některé servisní typy zpráv (redirect).

1.3.1. VELIKOST PAKETŮ

Maximální (teoretická) velikost IP paketu je 65535 B, ale limitujícím faktorem je většinou linková vrstva. Protože většina paketů aspoň jednou projde přes ethernet (nebo jeho ekvivalent), bývá většinou velikost paketů volena podle jeho limitu (1536 B), odtud nejobvyklejší hodnota 1500 B. To je ale samozřejmě pouze maximální hodnota, pakety často bývají i výrazně kratší, zejména u interaktivních aplikací. Hlavičky IP a TCP mají velikost 20-60 B (obvyklejší jsou hodnoty u dolní hranice), UDP a ICMP hlavičky mají 8 B, ethernetová hlavička 14 B (navíc 2 B na konci paketu kontrolní součet).



Vrstvy TCP/IP zajišťující přenos mezi dvěma hostiteli prostřednictvím dvou routerů.

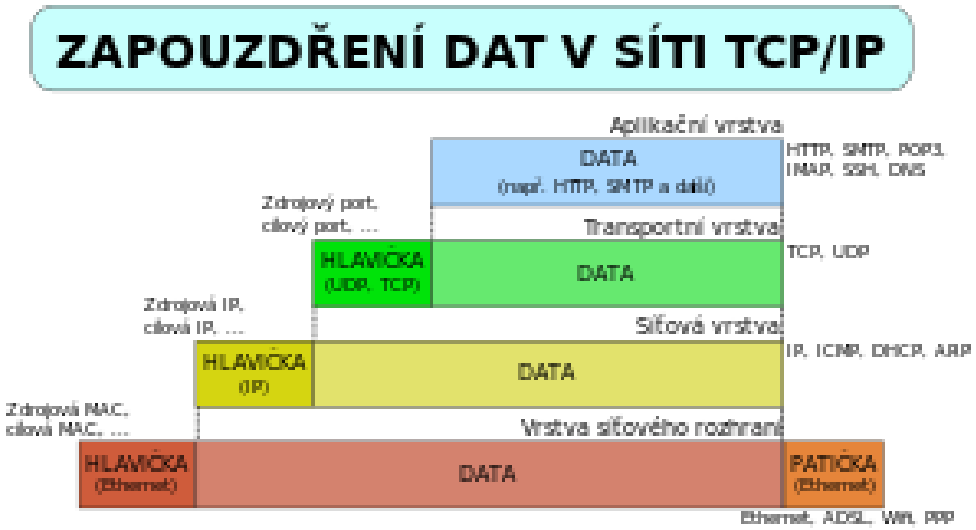


Schéma zapouzdření aplikačních dat na vrstvách TCP/IP.

Vzhledem ke složitosti problémů je síťová komunikace rozdělena do tzv. vrstev, které znázorňují hierarchii činností. Výměna informací mezi vrstvami je přesně definována. Každá vrstva využívá služeb vrstvy nižší a poskytuje své služby vrstvě vyšší.

Internet Protocol je základní protokol síťové vrstvy a celého Internetu. Provádí vysílání datagramů na základě síťových IP adres obsažených v jejich záhlaví. Poskytuje vyšším vrstvám síťovou službu bez spojení.

V současné době je převážně používán protokol IP verze 4. Nová verze 6, která řeší nedostatek adres v IPv4, bezpečnostní problémy a vylepšuje další vlastnosti protokolu

IP, je celosvětově používána jen několika procenty zařízení připojených k internetu, ale jejich počet rychle roste.

1.4. IPv4

- Internet protokol verze 4
- 32 bitové adresy
- cca 4 miliardy různých IP adres, dnes nedostačující
- formát: xxx.xxx.xxx.xxx kde xxx je libovolné číslo od 0 do 255 (8 bitů)

1.5. IPv6

- Internet protokol verze 6
- 128 bitové adresy
- podpora bezpečnosti
- podpora pro mobilní zařízení
- funkce pro zajištění úrovně služeb (QoS - Quality of Service)
- fragmentace paketů - rozdělování
- není zpětně kompatibilní s IPv4

Address Resolution Protocol se používá k nalezení fyzické adresy MAC podle známé IP adresy. Protokol v případě potřeby vyšle datagram s informací o hledané IP adrese a adresuje ho všem stanicím v síti. Uzel s hledanou adresou reaguje odpovědí s vyplněnou svou MAC adresou. Pokud hledaný uzel není ve stejném segmentu, odpoví svou adresou příslušný směrovač.

1.6. ICMP

Internet Control Message Protocol slouží k přenosu **řídících hlášení**, které se týkají chybových stavů a zvláštních okolností při přenosu. Používá se např. v programu *ping* pro testování dostupnosti počítače, nebo programem *traceroute* pro sledování cesty paketů k jinému uzlu.

1.7. TCP

Transmission Control Protocol vytváří virtuální okruh mezi koncovými aplikacemi, tedy **spolehlivý přenos dat**. Vlastnosti protokolu:

- Spolehlivá transportní služba, doručí adresátovi všechna data bez ztráty a ve správném pořadí.

- Služba se spojením, má fáze navázání spojení, přenos dat a ukončení spojení.
- Transparentní přenos libovolných dat.
- Plně duplexní spojení, současný obousměrný přenos dat.
- Rozlišování aplikací pomocí portů.

1.8. UDP

User Datagram Protocol poskytuje nespolehlivou transportní službu pro takové aplikace, které nepotřebují spolehlivost, jakou má protokol TCP. Nemá fázi navazování a ukončení spojení a už první segment UDP obsahuje aplikační data.

1.9. SCTP

Spolehlivý protokol pro přenos datagramů ve více proudech. Je využívaný zejména v telekomunikacích. Doplňuje některé vlastnosti, které TCP postrádá:

- Multihoming - komunikující uzel může mít několik IP adres.
- Členění datového toku na datagramy.
- Používání více proudů dat - omezuje blokování komunikace způsobené chybějícím blokem dat, ke kterému může dojít v TCP.
- Výběr a sledování cesty - Pokud má primární adresa problémy s dostupností lze používat alternativní.

2. Úvod do jazyka HTML

2.1. Úvod do HTML

HTML je jednoduchý značkovací jazyk, kterým se tvoří webové stránky, tyto stránky jsou jen obyčejné textové soubory, které většinou obsahují nějaký text a pár html značek, které určují význam a vzhled jednotlivých částí stránky.

Při tvorbě www stránek se nepoužívá jen html, ale i další jazyky: css, php, javascript a další.

HTML je základ, až budete znát alespoň základy můžete se přiučit něco o css stylech ty určují vzhled stránek a později něco z php, ale to už je programování.

2.2. Historie HTML

Web vznikl v roce 1989 od této doby se neustále vyvíjí, aktuálně se pracuje na HTML 5.

2.3. Co je potřeba pro tvorbu html stránek

Jednoduchý textový editor, může se použít poznámkový blok (notepad), nebo speciální editory s podporou html kódu, tyto programy zvýrazňují syntaxy a umožňují rychle přepínat mezi kódem a náhledem, práce s takovým editorem je mnohem efektivnější než s obyčejným poznámkovým blokem.

2.4. Používané termíny v HTML.

Tag Základní značka html, zapis tagu: <tag>.

Atribut Zapisuje se přímo do tagu a nastavuje nějakou jeho vlastnost, zápis atributu: <tag atribut="hodnota">.

Element Zápis nadpisu: <h1>Nadpis stránky</h1>.

První stránka.

HTML stránky jsou obyčejné textové soubory obohacené o tagy.

Tagy

Tagy se zapisují mezi znaky < >, některé jsou párové a některé nejsou.

Zápis nepárového tagu:

```
<tag>
```

Zápis párového tagu:

```
<tag>Nějaký text</tag>
```

U párového tagu je důležíte v ukončovací značce napsat lomítko / jinak by to prohlížeč nepochopil.

Atributy

Atributy se zapisují přímo do tagu.

```
<tag                                                    atribut="hodnota">
<tag atribut="hodnota">Párový tag s atributem.</tag>
```

Zákaz křížení tagů.

Tagy se mohou do sebe vkládat, nesmějí se však křížit.

```
<b><i>Tučná kurzíva</b></i>
```

Správný zápis:

```
<b><i>Tučná kurzíva</i></b>
```

Velikosti písmen

V html na velikosti písmen nezáleží můžete tedy psát **<TAG>**, nebo **<tag>** je to jedno, ale v xhtml což je vlastně novější verze html se už musí psát tagy i atributy jen malými písmeny.

Pozor v **url** je nutné zachovat velikost písmen, např. SOUBOR.html nerovná se soubor.html.

2.5. HTML

HTML znamená Hypertext Markup Language, tedy hypertextový značkovací jazyk. Hypertext markup language se vyvinul ze SGML a stal se používaným jazykem pro tvorbu webových stránek. V historii se nejvíce používaly verze HTML 2.0, HTML 3.2, HTML 4.01 a HTML 5. Z HTML se vyvinulo také XHTML (extended hypertext markup language) jako aplikace XML, které osobně považuji za slepou vývojovou větev (vývoj mi dal za pravdu). V roce 2010 se začalo mluvit o používání HTML 5 a větší část jeho novinek už se v roce 2017 používá.

Struktura html souboru

Nejčastější "šablona" stránky:

```
<!DOCTYPE HTML>
   <html>
            <head>
                  <meta charset="windows-1250">
                  <title>Jméno</title>
            </head>
            <body>

samotný text stránky

            </body>
   </html>
```

2.6. Jazyky pro prezentaci webového obsahu - CSS

2.6.1. HISTORIE CSS

CSS vzniklo někdy kolem roku 1997. Je to kolekce metod pro grafickou úpravu webových stránek. Ta zkratka znamená Cascading Style Sheets, česky "kaskádové styly". Kaskádové, protože se na sebe mohou vrstvit definice stylu, ale platí jenom ta poslední. To teď není důležité.

Existuje návrh CSS 2, vylepšené a složitější formy stylů, které ale v nejrozšířenějším prohlížeči Internet Exploreru moc nefungují.

Kdy používat CSS

V roce 2015 se dá říct, že už se celý web formátuje pomocí CSS. Ze starého HTML formátování zůstalo tak maximálně ztučnění a kurziva. Proto je dobré se v CSS orientovat, jestliže chcete dělat webové stránky. V první řadě je ale potřeba vědět, jak funguje HTML. Pokud HTML ani trochu neznáte, není dobré začínat s CSS. Kdy se vyplatí CSS studovat:

- chcete mít stránky hezky a moderně zformátované. Barvy, zarovnání, rozvržené sloupce atd.,
- často píšete texty určené pro Internet a nechcete ztrácet čas složitým formátováním,
- zabýváte se skriptováním, zejména Javascriptem,
- spravujete (či zatím jen plánujete) větší web s mnoha stránkami, které by měly vypadat podobně.

2.6.2. DALŠÍ MOŽNOSTI POUŽITÍ

Trojí použití CSS. Styl se může nadeklarovat třemi způsoby, níže uvádím příklady. Stačí, když se pro začátek naučíte jeden ze tří způsobů:

Přímo v textu zdroje u formátovaného elementu pomocí atributu style="...". Tomu říkám **přímý styl**. Je to nešikovné, ale občas se to používá.

Pomocí "**stylopisu**" (angl. "stylesheet") v hlavičce stránky. Stylopis je jakýsi seznam stylů. Je v něm obecně napsáno, co má být jak zformátováno, například že nadpisy mají být zelené. Do stránky se stylopis píše mezi tagy <style> a </style>.

Použitím externího stylopisu -- to je **soubor *.css**, na který se stránka odkazuje tagem <link>. V souboru je umístěný stylopis. Hlavní výhoda je v tom, že na jeden takový soubor se dá nalinkovat mnoho stránek, takže pak všechny vypadají podobně.

Samozřejmě stačí ovládnout jenom jeden způsob.

Příklady

Chci udělat odstavec červeným písmem pomocí CSS. Jak už jsem popsal, jde to třemi způsoby:

Přímý zápis

Do zdroje se napíše tato deklarace odstavce:

```
<p style="color: red">Tento odstavec bude červený.</p>
```

Vysvětlení: <p> je značka vymezující odstavec; z anglického paragraph. Atribut "style" je obecný atribut použitelný u každého prvku. Color znamená barva a red je červená.

Stylopisem

Do hlavičky dokumentu se napíše stylopis uzavřený mezi tagy <style></style>:

```
<style>
p    {color: red}
</style>
```

a do těla stránky se mohou psát odstavce:

```
<p>Tento odstavec bude červený. </p>
<p>Tento mimochodem také, protože červené budou všechny.</p>
```

To, jak zařídit, aby nebyly červené všechny, ale jenom některé odstavce, se dá pomocí "tříd" a "identifikátorů".

Externím CSS souborem

Vytvoří se soubor, který se pojmenuje třeba *styly.css*. V něm bude pouze tento text:

```
p {color: red}
```

Do hlavičky html dokumentu, který chci stylem ovlivnit, musím napsat odkaz na tento soubor:

```
<link rel="stylesheet" type="text/css" href="styly.css">
```

V těle dokumentu pak budou opět všechny odstavce červené.

2.6.3. SYNTAXE

CSS nejsou součástí HTML, a tak se zapisují zcela jiným způsobem, jak už jste si možná všimli. Pokud vám tabulka přijde příliš teoretická, všimněte si pouze příkladů ve spodní části.

Přímý styl:	<tag style="zápis vlastností">stylovaný element</tag>
Ve stylopisu:	<style> tag {zápis vlastností} 2.tag {zápis vlastností} </style>
Zápis vlastností zjednodušeně:	vlastnost: hodnota; 2.vlastnost: 2.hodnota
Zápis vlastností obecně:	vlastnost: hodnota [, hodnota2] [; další zápis vlastností]

Příklady:

Příklad přímého stylu	text červeného odstavce
Příklad stylopisu	<style> p {color: red} body {background-color: yellow;} </style>
Příklad jednoduchého zápisu vlastností	color: red
a složitějšího zápisu vlastností	font-family: Arial, Arial CE, sans-serif; color: red;

Je nutné všimnout si, kde se používají uvozovky, dvojtečky, složené závorky, středníky a čárky. Příklad správného zápisu:

```
h2 {color: green; background-color: yellow}
```

Mezery a konce řádků příliš velkou roli nehrají, mohou se přidávat a vypouštět. Velikost písmen nehraje roli. K dispozici je seznam vlastností a jejich hodnot.

Hodnoty, které prohlížeč nezná, ignoruje.

Komentáře ve stylopisech se dělají podobně jako v Javě mezi znaky /* a */. Dvě lomítka nefungují.

Příklad s nadpisem

Ve stylopisu nebo v externím css souboru se to dá udělat docela snadno.

```
<style>
    h1{color: green;}
    h2 {color: blue;}
</style>
```

Potom v celém dokumentu budou nadpisy první úrovně zelené a nadpisy druhé úrovně

modré. To ovšem pouze za předpokladu, že při psaní textu byly použity tagy <h1> a <h2>. Jinými slovy, stylopisy se dají efektivně použít pouze u dobře strukturovaných textů.

2.6.4. CSS STYLY

CSS styly jsou kaskádovací styly, používají se k vytvoření stylu webové stránky (barva, písmo, velikost písma). S CSS styly můžete pomocí jednoho souboru ovlivňovat design celého webu.

Zastaralé metody

Před CSS styly se k stylu stránky používal prvek <font face="písmo" color="barva" size="velikost">, který už byl překonán a zavrhnut. Oproti CSS stylu má nevýhody:

- Pokud jste často měnili styl textu, objevoval se tento tag ve zdrojovém kódu velmi často a to značně zpomalovalo běh stránky.
- Umožňuje měnit pouze písmo, barvu a velikost: <font face="písmo" color="barva" size="velikost">

2.6.5. DHTML

Možná už jste se někdy setkali se zkratkou DHTML, nebo Dynamické HTML - věřte, že tento jazyk je tvořen takřka jen z JavaScriptu, VBScriptu (jazyk s podobnými vlastnostmi jako JavaScript) a CSS styly. Tento jazyk využívá síly HTML, JavaScriptu a CSS a vytváří tak dokonalý design a stránky na které se dá dívat.

2.6.6. CSS STYLY A TŘÍDY, IDENTIFIKÁTORY A STYLE

CSS - Cascading Style Sheets - kaskádovací styly, poprvé implementovala společnost Microsoft v roce 1996 do Internet Exploreru 3.0. CSS styl zcela nahrazuje prvek <font> a zavádí prvek <style>. Pomocí CSS stylů můžete definovat kromě barvy, písma a velikosti spoustu dalších věcí (rámeček, podtržení, tučnost, vlnitost, zobrazení, odrážky, okraje..)

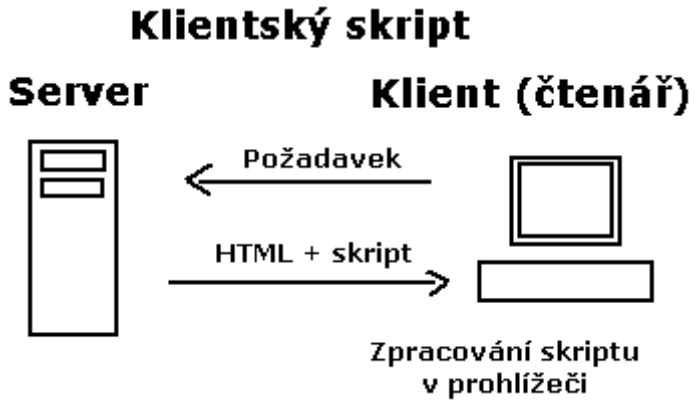
CSS styly se aplikují hlavně pomocí tříd a identifikátorů. Ty umožňují tvorbu CSS stylu jediným atributem a vy tedy nemusíte opakovat stejný kód desetkrát. Kromě toho můžete také definovat styl prvkům (h1, p, table at.) pomocí selektorů. Např. každý prvek <input> bude mít vždy červený text - to je možné udělat jediným řádkem CSS.

3. Logika na straně klienta - JavaScript

3.1. Co je JavaScript

JavaScript je programovací jazyk, který se používá v internetových stránkách. Zapisuje se přímo do HTML kódu, což je velká výhoda, protože je to jednoduché.

JavaScript je klientský skript. To znamená, že se program odesílá se stránkou na klienta (do prohlížeče) a teprve tam je vykonáván. (Protikladem klientských skriptů jsou skripty serverové, které jsou vykonávány na serveru a na klienta jdou už jen výsledky.)



Existují i jiné jazyky klientských skriptů, například VBScript. Jsou ale tak málo používané, že když se dnes mluví o "skriptech", myslí se tím JavaScripty.

JavaScript není Java

JavaScript je často zaměňován s Javou. Java je samostatný programovací jazyk. Má s JavaScriptem pouze podobnou syntaxi.

Co je potřeba umět

- HTML, alespoň základy
- základy programování, alespoň trochu

3.2. Charakteristiky jazyka

JavaScript je jazyk:

- interpretovaný -- nemusí se kompilovat
- objektový -- využívá objektů prohlížeče a zabudovaných objektů
- závislý na prohlížeči -- funguje ale ve většině prohlížečů
- case sensitivní -- záleží na velikosti písem v zápisu
- syntaxí podobný jazykům C, Java a podobným
- Omezení jazyka
- JavaScript funguje pouze v prohlížeči.
- Uživatel může JavaScript zakázat
- Existují různé odlišné verze jazyka i prohlížečů, což vede k častým chybám.
- Neumí přistupovat k souborům (kromě cookies) ani k žádným systémovým objektům.
- Neumí žádná data uložit (kromě cookies).
- To vše z něj dělá pouze jazyk druhořadý, účelově použitelný pouze v HTML stránkách.

Jak se učit JavaScript

Po zvládnutí základů je nejlepší všímat si cizích skriptů na cizích stránkách. Většina skriptů je zapsána přímo ve zdrojovém kódu stránky, takže se dá zkopírovat (některé kódy jsou v externích souborech, ale i ty jsou stáhnutelné).

3.2.1. VYSVĚTLENÍ SKRIPTU

Skript se zapisuje do HTML mezi tagy <script> a </script>. Všechno, co je mezi těmi tagy, je program psaný v jazyce Javascript.

V příkladě je použit příkaz **document.write()**. Ten způsobuje normální zápis do proudu dokumentu. Zapsaný text se ihned zobrazí v prohlížeči.

Pokud se zapisuje normální text, musí se obalit uvozovkami (na rozdíl od proměnné). Mezi uvozovkami se nesmí zalomit řádek.

Každý příkaz JavaScriptu se ukončuje středníkem nebo stačí zalomit řádek.

Jak vytvořit první skript

Vše, co vytvoříte v JavaScriptu se nazývá skript. Ten můžete volně umístit do stránky, nebo na něj vytvořit odkaz - stránka pak sama natáhne do stránky JavaScript. Samostatné soubory psané v JavaScriptu mají přípony .js nebo .jse. Přípona .js je obvyklejší. K vytváření skriptů stačí editor zdrojového kódu (PSPad, textový editor, nebo libovolný editor HTML). K prohlížení potřebujete prohlížeč prohlížeč (nejlépe alespoň Internet Explorer a Mozilla Firefox, abyste skripty mohli kontrolovat v těchto nejčastěji používaných

prohlížečích).

Vkládání skriptu do stránky

A teď prakticky. Skript píšeme mezi značky <script> a </script>. Ty můžete vložit do sekce body nebo do sekce head (záleží na účelu skriptu).

```
<html>
    <head>
                ...
                <script type="text/javascript">
                        javascript script body
                </script>
                ...
    </head>
    <body>

document body
    <script type="text/javascript">
                javascript script body
    </script>

document body
    </body>
</html>

Tag <script>
```

Syntax of the tag <script> is the following:

```
<script type="text/javascript" src="url of external file">
<!--
                    javascript script content
//-->
<script>
```

Atribut type označuje typ skriptu (v případě JavaScriptu "text/javascript"). Protože existují prohlížeče, které nemusí rozumět JavaScriptu, je vhodné zapsat na začátek skriptu <!-- a na konec //-->, jinak by došlo k tomu, že by prohlížeč vypsal skript jako normální text (nyní ho bude považovat za komentář a neukáže ho).

3.2.2. ZÁPIS SKRIPTU

JavaScript netoleruje záměnu velkého písmena za malé (je case-sensitive), proto document.write není to samé jako DOCUMENT.write. Toto pravidlo je nezbytné dodržovat, jinak skript nebude fungovat.

Tím jsme si prošli úvod do jazyka a můžeme se pustit do praktických příkladů. Když si jich pár vyzkoušíte, bude vše jasnější.

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, jehož autorem je Brendan Eich z tehdejší společnosti Netscape.

Nyní se zpravidla používá jako interpretovaný programovací jazyk pro WWW stránky, často vkládaný přímo do HTML kódu stránky. Jsou jím obvykle ovládány různé interaktivní prvky GUI (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků.

JavaScript byl původně obchodní název implementace společnosti Netscape, kde byl vyvíjen nejprve pod názvem Mocha, později LiveScript, ohlášen byl společně se společností Sun Microsystems v prosinci 1995 jako doplněk k jazykům HTML a Java. Pro verzi firmy Microsoft je použit název JScript. Ten je podporován platformou .NET.

Program v JavaScriptu se obvykle spouští až po stažení WWW stránky z Internetu (tzv. na straně klienta), na rozdíl od ostatních jiných interpretovaných programovacích jazyků (např. PHP a ASP), které se spouštějí na straně serveru ještě před stažením z Internetu. Z toho plynou jistá bezpečností omezení, JavaScript např. nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele.

JavaScript je možné použít i na straně serveru. První implementací JavaScriptu na straně serveru byl LiveWire firmy Netscape vypuštěný roku 1996, dnes existuje několik možností včetně opensource implementace Rhinola založené na Rhino, gcj, Node.js a Apache.

Kromě DHTML se JavaScript používá k psaní rozšíření pro mnohé aplikace, například Adobe Acrobat.

JavaScript je také možno spouštět v operačních systémech Windows pomocí programu Windows Script Host a nahradit tak dávkové soubory MS-DOS.

4. Architektura webu

4.1. Návrh architektury webu

Web nikdy nezačínáme tvořit grafickým návrhem. Na začátku nejprve definujeme cíle jednotlivých stránek a pak to, kde se jaké informace budou na webu nacházet.

Proč je to důležité?

Díky návrhu architektury webu si můžeme lépe představit, jak bude výsledný web vypadat a jaké budou jeho funkce.

Díky tomu můžeme lépe odhalit a odstranit případné nedostatky. Tím ušetříme spoustu peněz, které by mohla stát pozdější oprava naprogramované aplikace.

Návrh struktury webu

V návrhu struktury vydefinujeme všechny stránky webu, jejich vzájemné vazby a obvyklé uživatelské scénáře. Dbáme na to, aby byly informace vhodně strukturované, dobře použitelné a vedly k naplnění cílů.

Návrh navigace webu

Správně zvolená navigace je důležitou součástí webu, která pomáhá návštěvníkovy stránek rychle nalézt to, co hledá. Tomuto bodu věnujeme při návrhu webu velikou péči.

Rozkreslení modelů webu (wireframe)

Jakmile známe obsah webu, tak se můžeme pustit do kreslení drátěných modelů webu (wireframe), které jsou základem pro grafický návrh webu.

Jazyk PHP

PHP je jedním z nejvíce rozšířených programovacích jazyků používaných k vytváření webových aplikací. PHP se používá na straně serveru a slouží tedy ke generování HTML/XHTML kódu stránky, jenž pak server odesílá do prohlížeče (na rozdíl od klientského JavaScriptu, který funguje až při zobrazení stránky v prohlížeči).

Hlavním kladem PHP je jeho nezávislost na platformě (Windows, Linux, Unix…), mezi výhody PHP patří i široké možnosti použití. PHP například umí pracovat se soubory a s mnoha různými databázemi, s PHP lze generovat a upravovat grafiku, umí odesílat a přijímat emaily, vytvářet PDF, podporuje všechny důležité internetové protokoly…

Protože má PHP poměrně volnou syntaxi (způsob zápisu), snadno se učí, zejména pokud již máte zkušenosti s jinými programovacími jazyky. Společně s webovým serverem Apache a databází MySQL tvoří PHP tzv. triádu, trojici programů nejčastěji používaných pro generování stránek. Z toho plyne i další výhoda PHP – na internetu existuje obrovské množství fragmentů, uživatelsky definovaných funkcí a hotových řešení obvyklých problémů.

4.2. Programování webových aplikací

Jako **webová aplikace** se obvykle označuje skript (kód zajišťující funkci programu) běžící na straně serveru. Často bývá propojen s některou z databází, systému uchovávajícího data webové aplikace (zjednodušeně si lze databázi představit jako soubor MS Excel). Výstupem skriptu je potom samotná webová stránka, která je předána k zobrazení prohlížeči.

Úkolem webových aplikací je většinou zvýšit interakci internetové prezentace s jejími návštěvníky, případně též usnadnit správu webu, tj. ušetřit opakující se práci při tvorbě www stránek.

Podle požadavků na funkčnost může být webovou aplikací pouhých několik řádků kódu (např. při odesílání konkaktního formuláře), výjimkou však nejsou ani webové aplikace o mnoha tisících řádcích.

Složitější webové aplikace bývají často propojeny i na další software uvnitř firmy, např. na objednávkové systémy, účetní programy atd. To umožňuje ještě efektivněji šetřit drahou lidskou práci. Problémem nemusí být ani napojení aplikace na online platební systémy.

Příklady jednodušších webových aplikací

- Kontaktní formulář
- Kniha návštěv
- Diskuze nebo chat
- Katalogy a ceníky
- Různé slovníky
- Bannerový systém

U větších prezentací se často vyplatí řešit rovnou celou prezentaci dynamickým způsobem, ať už pomocí šablon či rovnou nasazením na redakční systém. Specifickou formou takové webové aplikace je potom internetový obchod. Pro své mnohé výhody se také stále více rozmáhají weblogy, taktéž různé intranety a extranety patří mezi speciální druhy webových aplikací.

Při programování všech webových aplikací dbáme především na tyto aspekty:

Bezpečnost – má prvořadý význam u jakékoliv webové aplikace, vždy hrozí riziko ztráty či zničení dat, nebezpečí číhá i v podobě krádeže informací nepovolanou osobou nebo v nabourání webového serveru skrze aplikaci (hrozba ztráty image společnosti).

Využívání dostupných zdrojů – při programování každé webové aplikace se snažíme používat již hotové úseky kódu z jiných zdrojů, k tomuto účelu vlastníme i rozsáhlý archív skriptů. Šetříme si tak práci potřebnou na vývoj aplikace a tím i vaše finance a celkový čas na realizaci zakázky.

Rozšiřitelnost – jakmile se webová aplikace osvědčí v praxi, začíná se obvykle pracovat na jejích dalších úpravách, vylepšeních a nadstavbách. Pokud bylo s těmito modifikacemi počítáno už při návrhu aplikace, jejich zapracování je mnohem jednodušší a tedy i levnější. I z tohoto důvodu se snažíme řešit většinu složitějších webových aplikací modulárním způsobem.

Rychlost – pomalá webová aplikace je jen velmi málo použitelná, problémy s ní mají i vyhledávače. Proto se snažíme optimalizovat všechny skripty na rychlost a proto také doporučujeme nasadit hotovou aplikaci na naše servery. Protože zde zároveň máme instalované nejmodernější vývojové nástroje, jaké na mnoha serverech pro komerční webhosting chybí, i samotný vývoj webové aplikace se tím urychlí a zlevní.

Maximální zátěž webové aplikace je pojem spojený s vysokou návštěvností. Pokud váš web navštíví současně větší množství návštěvníků (např. při uvedení očekávaného produktu), server je často nezvládne všechny obsloužit (říkáme, že server „spadl'). Schopnost webové aplikace odolávat vysoké zátěži vyžaduje především volbu vhodných nástrojů, optimalizaci databáze i výpočtů a další speciální techniky, jako je například *předkešování*. Podle našich zkušeností právě malá odolnost vůči zátěži mnohdy vypovídá o nízké úrovni programátorů, kteří webovou aplikaci vytvořili.

Důkladné testování – před spuštěním webové aplikace vždy důkladně testujeme všechny její funkce na vývojovém serveru. Ten mívá stejnou konfiguraci jako server ostrý, což umožňuje další redukci případných problémů při samotném spuštění.

4.3. Technologie našich webových aplikací

Při programování webových aplikací bývá dnes po celém světě nejčastěji využíván skriptový jazyk PHP a databáze MySQL. Tato kombinace, společně s webovým serverem (programem) jménem Apache nazývaná **triáda**, se i nám osvědčuje pro svou flexibilitu. Mezi další výhody této sestavy patří široká dostupnost hotových funkcí a fragmentů kódu a také neustálý vývoj těchto programů.

Jsou-li k tomu důvody z hlediska požadované funkčnosti webové aplikace nebo zjednodušení jejího vývoje, používáme i další programovací jazyky, např. Perl, Python či databázi PostgreSQL. Každý z nich se hodí na určité specifické potřeby konkrétní webové aplikace.

Největším kladem všech zmiňovaných technologií je jejich zařazení k open source. Znamená to, že jsou **šířeny zdarma** (což je také jedním z významných faktorů jejich dosavadní úspěšnosti a rozšířenosti) a vy tedy za samotné využívání těchto produktů nic neplatíte.

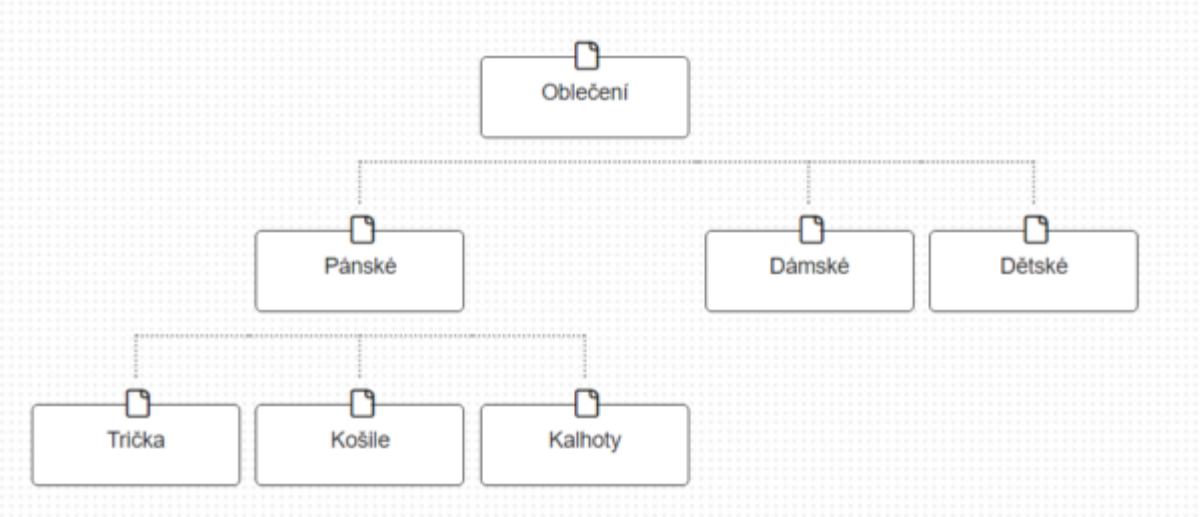
4.4. Co to je architektura webu?

Informační (obsahová) architektura webu je způsob, jakým poskládáte informace na webu do logického celku.

Je jasné, že všechny informace nebudete mít na jediné stránce, ale že web bude obsahovat více stránek. A tyto stránky jsou uspořádané do různých úrovní. Od toho obecného po podrobné. A vše je tematicky provázané.

Pokud například přijdete na web prodávající oblečení, tak se budete do webu zanořovat nějak takto: pánské oblečení > košile > s krátkým rukávem. Způsob, jakým jsou informace na webu uspořádané, se nazývá informační architektura webu.

Více naznačuje tento obrázek:



Proč řešit architekturu webu

Dobrá architektura webu je důležitá z několika důvodů:

- Vede návštěvníka webu rychle a intuitivně k cíli – tedy k obsahu, pro který si přišel.
- Tvoří logiku celého webu, kterou vnímají návštěvníci i vyhledávače.
- Umožňuje uspořádat obsah webu tak, aby byl snadno dohledatelný ve vyhledávačích.
- Jestliže připravujete nový web, nebo e-shop, obsahovou (informační) architekturu webu byste měli řešit v rámci základního návrhu webu (před grafickým návrhem a realizací webu – viz 4 kroky profesionálního webdesignu).
- Kvalitní architektura webu zajistí, že na něm budete mít správně uspořádaná

všechna důležitá témata. To se odrazí nejenom v lepší orientaci, ale také v lepší dohledatelnosti webu na konkrétní klíčová slova ve vyhledávačích. A jak je známo – návštěvnost z vyhledávačů je zadarmo a je vysoce relevantní.

4.4.1. JAK NA DOKONALOU ARCHITEKTURU

V rámci MD webdesign tvoříme architekturu webu na základě analýzy klíčových slov. To znamená, že zjistíme, jaká klíčová slova lidé používají ve vyhledávačích Google a Seznam. Analyzujeme jejich hledanost, tedy důležitost. A klíčová slova pak rozřadíme do kategorií, abychom získali větší nadhled.

Platí, že pokud chcete být na nějaké klíčové slovo dohledatelní ve vyhledávači, je vhodné mít na webu stránku věnovanou tématu klíčového slova a klíčové slovo samotné.

Díky analýze klíčových slov získáte přehled o tom, co uživatele vyhledávačů zajímá. A zařazením těchto témat (klíčových slov) do architektury webu vám výrazně zvýší šance na získání dalších pozic ve vyhledávání.

Když už vytvoříme informační architekturu, popíšeme i obsah jednotlivých stránek. Samotný obsah webu navrhujeme na základě analýzy zákazníka/klienta formou tzv. person (více o personách na Wikipedii). Díky tomu můžeme navrhnout takový obsah, který odpovídá očekávání konkrétních zákazníků. A jsme schopni mu dodat takové informace, o které stojí. A co nejdříve odbourat jeho/její obavy před provedením poptávky/objednávky.

Tvorba webových stránek: začínáme informační architekturou

Informační architektura je plán pro třídění informací na webu. Pro správnou funkci budoucích webových stránek je zapotřebí věnovat nemalé úsilí vytvoření srozumitelné informační architektury.

Stejně jako architekt při návrhu domu pracuje s prostorem, světlem a tvary, informační architekt pracuje s informacemi, strukturou a prioritou.

Obecně platí, že správně navržená informační architektura by měla být pochopitelná i "nahá", tedy tak, jak jsme ji vytvořili v textové podobě. Barvičky ani šipky návštěvníkovi při průchodu webem nepomohou, pokud informační architektuře webu nerozumí.

Mezi hlavní důvody "ztracení se návštěvníka na webu" patří: - použitá terminologie - uspořádání vyžadující profesní znalost - nelogické uspořádání informací.

Abychom se výše uvedeným problémům vyhnuli, měl by návrh informační architektury zahrnovat tyto kroky:

- Určení cílové skupiny
- Shromáždění informací
- Seskupení informací
- Určení priorit
- Vytvoření informační architektury

Kroky postupně:

1. Určení cílové skupiny

Je důležité si na otázku, kdo je cílová skupina webu, odpovědět dříve, než začneme navrhovat informační architekturu, protože dramaticky změní náš pohled na zveřejňované informace.

U webů veřejné správy je cílovým uživatelem prakticky každý, stěží proto budeme schopni definovat znalosti nebo zvyky, které bude většina uživatelů sdílet. Můžeme ale předpokládat, které znalosti uživatelé mít nebudou.

Praxe ukazuje, že uživatelé, kteří přistupují na webové stránky měst a obcí, zpravidla neznají rozdíl mezi městem a úřadem. Neznají úřední terminologii a stěží se vyznají v organizační struktuře úřadu. Situaci nepomáhá skutečnost, že totožné agendy jsou v různých městech spravovány různě nazvanými odbory.

Plyne z toho, že interní struktura úřadu není vhodnou inspirací pro tvorbu informační architektury, kterou chcete prezentovat laické veřejnosti.

2. Shromáždění informací

Druhým krokem při návrhu informační architektury by mělo být ujasnit si, jaké informace bude návštěvník na webu hledat. Veřejná správa musí řadu informací zveřejňovat ze zákona, ale zdaleka to nejsou všechny informace, které by se na webu měly vyskytovat.

Weby veřejné správy zpravidla obsahují obrovské množství informací, od zápisů jednání zastupitelstev, přes vyhlášky, rozpočty, nařízení, rozhodnutí až po aktuální dění v obci a nabídku kulturních či sportovních akcí.

Součástí tohoto kroku je i rozhodnutí, které informace na webu zveřejňovat nechceme. Je skutečně nutné na webu úřadu zveřejňovat kalendář akcí nebo seznam místních restaurací? Vždy je třeba zvážit, které informace bude provozovatel webu schopen pravidelně aktualizovat. Problémům s neaktuálními informacemi je dobré předejít již při návrhu informační architektury.

3. Seskupení informací

Když víme, které informace chceme na webu prezentovat, je na čase vytvořit tematické celky. Tyto celky by měly pojmout veškeré informace, které jsme nashromáždili v předchozím kroku, a to pokud možno bez přesahů.

V tuto chvíli je vhodné do procesu zapojit alespoň jednoho zástupce laické veřejnosti. Jako pracovník úřadu máte jasno ve struktuře úřadu a v procesech, kterými se řídí. Pro laika je však tento svět cizí a je tedy nepraktické po návštěvníkovi pro úspěšný průchod webem požadovat vnitřní znalost úřadu.

Návštěvník postupuje webem “shora” – začne na nejobecnější úrovni a postupně pomocí dalších klikání svůj dotaz upřesňuje, až dosáhne konkrétního výsledku. Při návrhu informační architektury však postupujeme “zdola” – třídíme a kategorizujeme jednotlivé

informace do celků a skupin. Tento rozpor může velmi snadno návštěvníka zavést do slepých uliček. Návštěvník nezná obsah webu a neví, zda hledaná informace na webu vůbec existuje. Je proto klíčové otestovat srozumitelnost informační architektury při průchodu "shora".

4. Určení priorit

Doposud jsme si vystačili s trochou selského rozumu a testování s uživateli. Při určení priorit v návrhu informační architektury se však pouštíme do analytické části a je dobré mít v ruce konkrétní čísla, například analýzu nejčastěji vyhledávaných výrazů na webu. Priority se navíc mohou měnit v čase, například v létě bývá častý dotaz na koupaliště, zatímco v zimě uživatele zajímá především úklid sněhu.

Způsobů upřednostnění informací je celá řada: informaci můžeme umístit na přední místa ve struktuře, případně umístit ji přímo na titulní stránku nebo hlavní stránku tematického celku. Dále lze použít vizuální zvýraznění za pomoci kontrastu, barev, velikosti atd.

Například na stránce komunálního odpadu by mohla být informace o aktuální výši a datu splatnosti poplatku, aniž by návštěvník byl nucen procházet PDF s vyhláškou. A protože se jedná o často hledanou informaci, můžeme na ni navíc upozornit větším fontem.

5. Vytvoření informační architektury

Při vytváření informační architektury může pomoci vhodný nástroj, ale žádný nástroj není samospásný. Informační architektura ve své nejjednodušší podobě připomíná strom.

5. PHP /basics/

PHP is a programming language working on the side of server. PH/ enables to store and change website data. Původní význam zkratky PHP byl Personal Home Page. Vzniklo v roce 1996, od té doby prošlo velkými změnami a nyní tato zkratka znamená PHP: Hypertext Preprocessor.

Možnosti PHP

PHP není nijak těžké pochopit a už se základy si lze vystačit. Umí ukládat, měnit a mazat data. Vše se odehrává na webovém serveru (kde jsou uloženy zdrojové kódy webových stránek). PHP skript se nejprve provede na serveru a potom odešle prohlížeči pouze výsledek (znamená to, že nejprve spočítá kolik je 300/30 a pak prohlížeči odešle jen číslo 10). Proto ve zdrojovém kódu najdete jen "10" (to je rozdíl oproti JavaScriptu, který počítá přímo v prohlížeči). Zdrojový kód PHP narozdíl od JavaScriptu a HTML v prohlížeči nezobrazíte.

Pomocí PHP je možné vytvořit diskuzní fórum, knihu návštěv, počítadlo, anketu, graf a dokonce si pomocí jednoduchého kódu můžete zlikvidovat celý obsah webu. Navíc máte možnost propojit vaše stránky s databázemi, např. MySQL.

K čemu PHP?

Je rozhodně alespoň jedna funkce PHP, která se hodí snad do každého webu. Na webových stránkách se obvykle opakují některé části, hlavička s odkazy, menu, patička. S PHP si můžete snadno vytvořit šablonu pro web, do které se budou vkládat soubory s menu, patičkou atd. Můžete tedy mít menu jen jednou zapsané a do dalších stránek ho pouze kopírovat. Až budete chtít menu změnit, bude to nesmírně jednoduché. Více v článku PHP menu.

Soubory PHP

Webová stránka s prvky PHP má nejčastěji koncovku .php. Lze se však setkat i s dalšími koncovkami, např. .phtml, php3, php4, php5. Některé hostingy dle koncovky určovaly, pod jakou verzí PHP skript spustit (aktuální je 7). To je však velice výjimečné a v naprosté většině případů byste si měli vystačit s koncovkou .php.

Instalace

PHP je jazyk, který si nevystačí jen s prohlížečem určité verze (třeba jako HTML nebo JavaScript), ale je nutné ho na počítač nainstalovat. Základ tvoří webový server a knihovny. K podpoře PHP je třeba instalovat a konfigurovat server, obvykle Apache. Nejlepší je využít k instalaci PHP program PHP Triad, který vše sám nainstaluje.

Webhosting s PHP

Ne každý webhosting zahrnuje podporu PHP. Potřebná podpora je u webhostingu nadstandardní službou za příplatek. Nicméně lze sehnat webhosting zdarma s podporou PHP (např. Webzdarma.cz, PHP 5). Při výběru webhostingu pro PHP stránky si pečlivě

přečtěte, co nabídka zahrnuje.

5.1. PHP – základní informace

Dynamické stránky (tj. stránky nejprve vygenerované serverem a poté odeslané klientovi) jsou v současné době nezbytnou součástí každé složitější internetové prezentace. K hlavním skriptovacím jazykům, které se používají k tvorbě těchto stránek, patří ASP (Active Server Pages) a PHP. Právě úvodu do jazyka PHP je věnován tento seriál.

Před časem již na Intervalu úvodní článek o PHP vyšel, takže zde je jen přehled základních informací o tomto jazyce:

Co to je PHP?

PHP je skriptovací jazyk vykonávaný na straně serveru vkládaný do běžného HTML kódu. Co to znamená? Každou stránku, která obsahuje PHP skripty, server nejprve vezme a vykoná všechny příkazy v PHP, které jsou ve stránce uvedené, poté pošle klientovi již čistý HTML kód, který je výsledkem běhu skriptu. Server může PHP skripty teoreticky hledat ve všech odesílaných souborech, ale zpravidla je nakonfigurován tak, aby je hledal v souborech s příponami .php, .php3 nebo .phtml. Příkazy PHP jsou vkládány přímo do HTML kódu a jsou od něj odděleny tagy <? a ?> (nebo <?php a ?>).

K čemu je PHP dobré?

PHP je velmi všestranný jazyk, ve kterém lze relativně snadno naprogramovat třeba zpravodajský server nebo virtuální obchod. Data si lze ukládat buď do obyčejných textových souborů, nebo do databáze (PHP si dobře rozumí s většinou běžně používaných databází, jmenujme alespoň populární mySQL). Hračkou je zpracovávání dat z formulářů, snadno vytvoříte různé on-line testy včetně statistik úspěšnosti dosavadních návštěvníků nebo naprogramujete kvalitní reklamní systém. O síle PHP svědčí jeho používání na serverech Email.cz, Centrum.cz či BillBoard.cz

Co je k tvorbě v PHP potřeba?

PHP se vkládá do HTML, takže k tvorbě PHP skriptů obvykle postačuje jakýkoliv běžný HTML editor. Já osobně při tvorbě jak HTML, tak PHP skriptů zcela vystačím s Poznámkovým blokem (Notepadem). Nejdůležitější je ale mít vytvořené skripty kde umístit a vyzkoušet. K tomu musí být na serveru nainstalována podpora PHP (PHP – nyní ve verzi 4 – je zdarma ke stažení na www.php.net, zde je i podrobný postup instalace). Nejvýkonnější je PHP jako modul serveru Apache pod operačním systémem Linux, ale lze ho používat i pod Windows – tato informace vás samozřejmě zajímá jen tehdy, pokud se staráte o svůj server sami. Pokud používate některou z mnoha nabídek webhostingu zdarma, není podpora PHP moc pravděpodobná, nicméně např. server www.kgb.cz webhosting zdarma včetně podpory PHP (byť s jistými omezeními – nemožnost používání databází atd.) poskytuje. Máte-li placený webhosting, zeptejte se

na PHP svého providera – je slušná šance, že podporu PHP buď zdarma, nebo za určitý příplatek nabízí. Provider by vám také měl sdělit, jakou příponu máte požívat pro soubory obsahující PHP skripty (jak již bylo řečeno, většinou je to .php, .php3 nebo .phtml).

První skript v PHP

Zde je náš první PHP skript, který (jak originální :-) vypíše aktuální čas:

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
        <title>PHP – ukázka 1</title>
    </head>
    <body bgcolor="#FFFFFF" text="#000000">
        <center><font face="Arial CE, Arial" size="5">
            Aktuální čas: <?php echo Date („H:i:s"); ?>
        </font></center>
    </body>
</html>
```

Uložte ho na server třeba jako soubor ukazka1.php a poté se na něj podívejte přes nějaký prohlížeč. K tomu, aby skript pracoval, musí být nejprve interpretován serverem, tzn. nelze si ho prohlížet off-line ze svého harddisku.

Jak skript přesně pracuje?

Důležité je si všimnout, že jde v podstatě o klasickou HTML stránku, která navíc obsahuje jeden PHP příkaz, a to echo Date („H:i:s"), který je od okolního HTML kódu oddělen značkami <?php a ?>. Server nejprve vezme požadovaný soubor ukazka1.php, vidí, že má příponu .php, a proto ho nejprve prožene interpretem PHP a vykoná všechny příkazy – ty hledá právě mezi oddělovacími značkami <?php a ?>. Narazí na příkaz echo, který slouží k zápisu do výsledného souboru. Funkce Date vrací datum a čas, v závorce jsou její parametry, které v tomto případě určují formát, ve kterém má být čas zobrazen (hodiny, minuty a sekundy oddělené dvojtečkami – více o funkci Date v dalších dílech). Následující

středník slouží k oddělení více příkazů od sebe – v tomto případě je nadbytečný, protože příkaz je jen jeden, ale je dobré zvyknout si ho psát. Interpret PHP tedy namísto příkazu PHP zapíše do stránky aktuální čas a server poté stránku odešle návštěvníkovi. Pokud si dáte zobrazit zdroj výsledku skriptu, opravdu uvidíte pouze čisté HTML, žádné PHP. Z toho vyplývá další hezká vlastnost PHP – na rozdíl od client-side jazyků, jako je třeba JavaScript, se k vašemu pracně vytvořenému kódu nikdo nedostane a nemůže ho tak lacino okopírovat.

Shrnutí anebo co si je třeba pamatovat

- Příkazy PHP se vkládají do běžného HTML kódu, oddělují se značkami <?php a ?> (nebo pouze <? a ?>).
- Aby server věděl, že má v souboru hledat příkazy PHP, musí mít soubor správnou příponu, obvykle .php, .php3 nebo .phtml.
- Návštěvník od serveru dostane pouze čistý HTML kód (tedy kód po vykonání všech PHP příkazů).

6. Zpracování http dotazu

6.1. Metody dotazu

GET

je nejpoužívanější metoda. Slouží k vyzvednutí objektu (html soubor, obrázek, cokoliv...) ze serveru. Odpověď je „kešovatelná“. Proto GET dotaz doprovází spoustu hlaviček ve kterých se specifikuje, jak je dokument starý, zda byl modifikován atd. GET dotaz obvykle nemá tělo.

POST

Pomocí této metody se dají v těle dopravit na server informace od uživatele (velmi často se POST používá pro odeslání rozsáhlejších dat z webových formulářů, pro upload souboru a podobně).

HEAD

se chová naprosto stejně jako GET, ale v odpovědi se nepřenáší tělo. Tento dotaz se hodí například ke zjištění, zda objekt existuje (při kontrole odkazů na stránce).

PUT/DELETE

vytvoří/smaže daný objekt ze serveru. Tyto metody se v praxi příliš nevyužívají.

OPTIONS

slouží ke zjištění informací o daném kontextu (nebo „*“ pro celý server). Klient může zjistit, které dotazy může na daný kontext zaslat.

OPTIONS * HTTP/1.1

Host: www.root.cz

Ukázka implicitního nastavení serveru

TRACE

se používá ke sledování cesty celého dotazu. V těle odpovědi klient dostane pěkně seřazené všechny dotazy jednotlivých systémů, kterými požadavek procházel. Tato metoda je používaná administrátory a webovými programátory, kteří chtějí zjistit, proč jim server vrací například prošlý (expirovaný) dokument apod.

Hlavičky

Protokol HTTP verze 1.1 definuje velké množství hlaviček pro dotazy i odpovědi. Zde jsou některé z nich:

6.1.1. DOTAZOVÉ HLAVIČKY

Accept*

Hlavičky tohoto typu indikují, co všechno je schopen klient zpracovat. Server pak vybere nejvhodnější alternativu. Patří sem hlavičky Accept (MIME typy dokumentů), Accept-Charset (znaková sada, v českém prostředí velmi důležité), Accept-Encoding (kódování přenášených dat, většinou slouží k výběru komprese) a Accept-Language (jazyk dokumentu).

Connection

V protokolu HTTP 1.1 je definován parametr „close“, který požaduje okamžité uzavření spojení po přenosu prvního vyžádaného dokumentu.

Referer

Klient touto hlavičkou sděluje URI stránky, ze které byl odkaz vygenerován.

Host

HTTP 1.1 zavádí podporu tzv. name-based virtuálních serverů. Tato metoda umožňuje provozovat více virtuálních serverů na jediné IP adrese, klient ovšem musí pomocí této hlavičky specifikovat jméno serveru, s nímž chce komunikovat.

User-Agent

Touto hlavičkou by se měl klientský program identifikovat, ať už pro účely statistické či pro poskytování odlišného obsahu různým prohlížečům a podobně.

6.1.2. HLAVIČKY ODPOVĚDI

Content*

Hlavičky popisující obsah (tělo) odpovědi. Mohou obsahovat například délku obsahu (Content-Length), jeho MD5 digest (Content-MD5), jazyk (Content-Language), typ dokumentu (Content-Type) a další atributy. Nutno podotknout, že tyto hlavičky se nepoužívají pouze v odpovědích – pakliže obsahuje požadavek i tělo (např. při metodě POST), je obvykle nutné je rovněž použít.

Server

Tato hlavička slouží serveru k vlastní identifikaci (obvykle zde najdeme jeho jméno, verzi a někdy i další informace).

Expires

Server může prostřednictvím tohoto údaje sdělit, kdy vyprší platnost dokumentu. Po

uplynutí této doby by si měl klient stáhnout novou verzi.

Existuje i řada dalších hlaviček, kterými může například klient řídit stažení dokumentu („stahuj pouze pokud byl dokument modifikován od …“) nebo třeba předat serveru uživatelské jméno a heslo pro přístup k neveřejným částem serveru. Podobně i server může jemněji popsat svou odpověď a například sdělit klientovi, kdy byl naposledy dokument modifikován nebo zda je povoleno jeho kešování ve veřejných či privátních keších.

6.2. Základní rysy protokolu HTTP

K úplnému pochopení článku budeme muset znát už trochu více základní rysy HTTP, a jak tento protokol vlastně pracuje (znalost předchozího článku také pomůže). Protokol HTTP je protokolem aplikační úrovně pro distribuované hypermediální informační systémy. V praxi to znamená, že tento protokol se obecně na internetu používá nejen pro přenos dat mezi klientem a serverem, ale i pro mnoho dalších úloh. Protokol HTTP je bezstavový, tj. že nerozpoznává klienty, od nichž chodí požadavky. Pokud jeden klient odešle požadavek a vzápětí ten stejný klient odešle další požadavek, server nepozná, že jde o stejného klienta.

HTTP existuje ve 3 verzích a to 0.9, 1.0 a 1.1 . První z nich, označována za HTTP/0.9 existovala jako jednoduchý protokol, který uměl v omezené podobě přenášet data na internetu. Verzí HTTP/1.0 nabyl protokol možnosti přenášení informací ve formátu MIME, takže mohl obsahovat i metainformace o přenášených datech. Nejpodstatnějším vylepšením protokolu verzí HTTP/1.1, což je zároveň poslední, aktuální verze, bylo to, že všechna spojení se stala trvalými. To znamená, že se spojení uzavře, až když jeden z dvojice klient-server odešle hlavičku pro uzavření. Dříve HTTP uzavíralo spojení po každé odpovědi serveru. Tímto vylepšením se také nesrovnatelně zvýšila rychlost přenosu, protože server už pro každý obrázek, rám a applet nemusí otevírat nové spojení.

6.2.1. FORMÁT POŽADAVKU PROTOKOLU HTTP

Požadavek protokolu HTTP má následující formát:

METODA URL-DOKUMENTU VERZE-HTTP
HLAVIČKY
prázdný řádek
DALŠÍ-DATA Pouze u metody POST

Metoda požadavku značí způsob, jakým má server požadavek zpracovat. O metodách si více povíme později. Hlavičky se odesílají v následujícím formátu:

JMÉNO-HLAVIČKY: HODNOTA-HLAVIČKY

Každá hlavička musí být na samostatném řádku. Všechny řádky musí být ukončeny

sekvencí znaků CRLF (\r\n). Na konci všech hlaviček musí následovat prázdný řádek, i kdyby za ním už neměla být žádná data.

Požadavky se v PHP odesílají přes tzv. *sockety*. Socket je v podstatě jakési spojení mezi serverem a klientem. Abychom s nimi mohli pracovat, musíme takový socket nejdříve otevřít. K tomu slouží funkce fsockopen:

```
fsockopen(server, port);
```

Příklad:

```
$sock = fsockopen("www.interval.cz", 80);
```

Socket máme otevřený, můžeme odeslat svůj požadavek pomocí funkce fputs:

```
fputs(socket, request);
```

Příklad:

```
fputs($sock, "GET /index.html HTTP/1.1\r\nHost:
www.interval.cz\r\n\r\n")
```

Více v PHP již není třeba, pojďme se podívat na metody požadavku http.

7. Metody požadavku protokolu HTTP

V HTTP/1.1 existuje sedm základních metod požadavků HTTP:

- GET
- POST
- HEAD
- OPTIONS
- PUT
- DELETE
- TRACE

Za každým požadavkem ještě mohou následovat jednotlivé hlavičky. Ve verzi HTTP 1.1 je v každém požadavku povinná hlavička Host, která specifikuje hostitele. Po hlavičkách (jak už jsem psal) musí následovat prázdný řádek.

7.1. Metoda GET

Metoda GET je ta nejjednodušší a patří mezi základní. Prakticky pokaždé, když načítáte stránku ze serveru a neodeslali jste předtím formulář s metodou POST, používá se k obdržení stránky ze serveru právě tato metoda. Výsledkem je tudíž stránka a její hlavičky, na kterou se pomocí metody ptáme. Formát této metody je následující:

GET URL-STRÁNKY VERZE-PROTOKOLU
HLAVIČKY
prázdný řádek

Příklad:

GET /index.asp HTTP/1.1
Host: www.interval.cz
prázdný řádek

7.2. Metoda POST

Metoda POST funguje v podstatě sejně jako metoda GET, ale u POST máte možnost za hlavičkami (a prázdným řádkem) poslat skriptu data. Touto metodou se např. odesílají data z formuláře s metodou POST. Formát tohoto požadavku je následující:

POST URL-STRÁNKY VERZE-PROTOKOLU
HLAVIČKY
prázdný řádek
DATA Z FORMULÁŘE

Příklad:

POST /zpracujdata.php HTTP/1.1
Host: www.formulare.cz
Content-Length: 29
Content-Type: application/x-www-form-urlencoded
prázdný řádek
pole1=hodnota1&pole2=hodnota2

U této metody nám přibyly některé hlavičky, o kterých jsem se předtím nezmínil. Content-Length značí, jak dlouhá jsou data z formuláře (v bytech) a Content-Type: application/x-www-form-urlencoded značí MIME typ dat z formuláře.

Metody HEAD, OPTIONS, PUT, DELETE, TRACE

Pokud zrovna neprogramujete internetový prohlížeč (a v PHP asi ne), pravděpodobně se s těmito metodami nikdy nesetkáte. Takže pouze stručně uvedu jejich význam v tabulce:

Metoda	Popis
HEAD	Metoda HEAD funguje prakticky stejně jako GET, ale HEAD nevrací tělo stránky, pouze hlavičky. Toho se např. využívá při zjišťování, jestli se stránka od posledního požadavku změnila.
OPTIONS	Používá se pro dotaz na možnosti serveru
PUT	Funguje jako GET, ale uchovává tělo požadavku na místě daném požadovaným URL. Podobně jako odesílání souborů přes FTP.
DELETE	Odstraňuje dokument ze serveru. Dokument, který má být odstraněn, je dán URL požadavku.
TRACE	Používá se pro sledování požadavku přes všechny proxy servery a firewally, přes které požadavek jde. Podobá se nástroji TraceRoute.

7.3. Použití datových zdrojů

Moderní a výkonné aplikace pro zpracování dat neukládají informace přímo ve svých vlastních souborech, ale používají služeb některého z externích datových zdrojů. Datových zdrojů může být celá řada. Od jednoduchého textového souboru, přes dříve oblíbené souborové databáze třeba ve formátu DBF až po databáze uložené na SQL serverech.

Každý takový zdroj zpravidla používá svůj vlastní formát pro uložení dat a metody pro přístup k těmto datům. Aby programátoři aplikací nemuseli čelit problému s různorodostí datových zdrojů a vyvíjet aplikace pro každý zdroj zvlášť, byl vytvořen standardní nástroj, který umožňuje přistupovat k různým zdrojům dat pomocí jednotné standardizované platformy – ODBC.

Takto standardizovaná platforma přináší uživatelům možnost komplexně a přitom jednoduchým způsobem využívat data z různých zdrojů na různých místech. Můžete například přímo ve Wordu nebo Excelu použít údaje pořízené v Accessu, aniž by bylo nutné je nejprve exportovat a pak složitě načítat do požadované aplikace.

7.4. Co je ODBC

ODBC je zkratka pro Open Database Connectivity. Datové zdroje ODBC jsou aplikacím přístupné přes příslušný ovladač, který si můžeme představit jako prostředníka pro komunikaci mezi uživatelskou aplikací a externím zdrojem dat. Aplikace svůj dotaz na data předá do ODBC. Příslušný ovladač přeloží tento dotaz tak, aby mu rozuměl externí zdroj dat, a zašle mu jej. Odpověď od zdroje dat opět putuje přes ODBC, které přeloží výsledek do standardní podoby a vrátí jej volající aplikaci.

Princip práce ODBC je dán standardem, proto jakákoliv aplikace, která umí použít ODBC ovladače, může přistupovat k jakémukoliv externímu zdroji dat od libovolného výrobce, který pro to poskytne příslušný ovladač. ODBC tak umožnil standardizovat na straně aplikací přístup k datům a nestarat se o to, jakým konkrétním způsobem pracuje zdroj dat. Na jedné straně tedy aplikaci stačí, když umí pracovat s ODBC. Na straně druhé výrobci různých zdrojů dat k nim poskytují příslušné ODBC ovladače, aby jejich zdroje byly lehce přístupné aplikacím. To je výhodné pro programátory aplikací, kteří mají k dispozici nezávislý standardizovaný přístup k datům, a potažmo i pro uživatele, kteří tím pádem získají aplikace přistupující k různým datům levněji a v kratším čase.

Práce s ODBC ve Windows

Ovládací prvky pro nastavování ODBC jsou již integrální součástí operačního systému Microsoft Windows. **Správce zdrojů dat ODBC** naleznete v ovládacích panelech. Na tomto místě je nutno upozornit, že uvedené postupy i obrázky platí beze zbytku v operačním systému Windows 10, ale obdobně platí i pro další verze Windows. Na některých systémech ale může být správce ODBC přístupný odlišným způsobem.

8. Datové zdroje

K naplnění seznamů příjemců a šablon těmi správnými daty ve správný okamžik využívá Mailkit podporu pro **XML & RSS datové zdroje**. Jejich použití je omezeno v závislosti na typu účtu uživatele. Verze Mailkit Base je omezena na použití XML zdrojů dat, určených pouze pro import příjemců do seznamů, zatímco Mailkit Syndicate a Agency podporuje jak XML, tak RSS datové zdroje, a to i za účelem načítání dat do šablon.

Použití XML datových zdrojů pro seznam příjemců

Pro vytvoření nového seznamu příjemců z datového zdroje je nejprve potřeba nastavit nový XML datový zdroj.

- Jméno - jméno datového zdroje. Pokud bude datový zdroj sloužit pro seznam příjemců, dostane seznam příjemců shodné jméno.
- Popis - popis datového zdroje.
- Zdroj - URL adresa, na které se nalézá XML, či RSS, jež má sloužit jako datový zdroj.
- Autorizace - zaškrtne se v případě, že je přístup na umístění datového zdroje zaheslován.
- Typ - z roletkové lišty se vybírá, zda se jedná o RSS či XML datový zdroj
- Cíl - z roletkové lišty se vybírá cíl datového zdroje. Zda bude použit pro seznam příjemců nebo pro šablonu.
- Automaticky aktualizovat - pokud se tato volba zaškrtne, bude se zdroj automaticky aktualizovat těsně před rozesláním kampaně.
- Doba expirace - po jak dlouhé době se má datový zdroj v případě automatické aktualizace znovu aktualizovat.
- Poslední aktualizace - zde se zobrazuje datum poslední aktualizace datového zdroje.
- Prázdné záznamy: Vynulovat - údaje u příjemců budou vynulovány dle prázdných záznamů v importu. Ponechat současnou hodnotu - údaje zůstanou u příjemce ve stejné podobě jako před importem.

Jak připravit datový zdroj

Datový zdroj nemá pevně definovanou strukturu a může být ve formátech XML nebo JSON, který musí být plně validní. Soubor datového zdroje musí být vystaven na URL dostupné ze serverů Mailkitu a zabezpečen proti přístupu třetích stran neboť se jedná o citlivé údaje.

Protože každý klient využívá jiný informační systém s jinými možnostmi, systém datových zdrojů je postaven maximálně univerzálně a nepředepisuje specificky strukturu požadovaných dat. Na datové zdroje se však vztahují jistá technická omezení:

- Plně validní XML nebo JSON
- U XML formátu nejsou podporovány attributy (např. first_name="Jana" gender="f" country="cz")
- Doporučené kódování znaků UTF8
- Nepoužívat znak "," jako oddělovač pro více hodnot, ale použít znak "|"
- Unikátním identifikátorem záznamu a zároveň jediným povinným polem je e-mailová adresa. Pokud bude více záznamů s totožnou e-mailovou adresou, dojde k jejich vzájemnému přepsání.

Jak již bylo napsáno, jediným povinným údajem je **e-mail** a všechny další údaje jsou nepovinné, nicméně důležité. Obecně platí pravidlo "čím více, tím lépe", ale také "nic se nesmí přehánět". Do datového zdroje by tak mělo přijít maximum dostupných údajů o příjemcích, které je možné pro vaše současné, ale i budoucí e-mailové kampaně využít.

Import dat z datového zdroje

K získání obsahu se musí přiřadit jednotlivé větvě zdroj k polím kontaktu. Pro přiřazení se klikne na název zdroje dat a dále na tlačítko **Zobrazit strukturu**. Po přiřazení všech polí, se klikne na tlačítko **Uložit**. Následně bude možné pokračovat tlačítkem **Import** pro aktuální import dat. V tomto momentě bude vytvořen nový **Seznam příjemců** (jež bude pojmenován totožně jako zdroj, jejž ho vytvořil) a data z datového zdroje budou importována dle Vašeho předchozího přiřazení.

Uživatelé účtu typu Syndicate a Agency mají zároveň možnost automatické aktualizace. Pokud označí automatickou aktualizaci, datový zdroj bude automaticky importován a seznam příjemců bude aktualizován před počátkem doručování každé kampaně. Tento parametr nedoporučujeme používat u datových zdrojů obsahující více než nižší tisíce záznamů, neboť samotná aktualizace může trvat i několik minut a o to bude zpožděna rozesílka kampaně. Pro větší datové zdroje doporučujeme využít možnosti pravidelné plánované aktualizace ve stanovenou denní hodinu, kterou pro vás nastaví naše zákaznická podpora.

Použití XML & RSS datových zdrojů v šablonách

Nastavení XML & RSS datových zdrojů pro použití v šablonách je velmi podobné jako pro použití v seznamech příjemců, ale bez nutnosti přiřazování významů jednotlivých polí. Hodnoty jsou určeny řetězcem jmen v šabloně, proto je snadné nastavit jakýkoliv XML či RSS zdroj.

Výše je uvedený příklad kódu šablony, pro kterou je použit RSS datový zdroj pojmenovaný *EXAMPLE*. Příkaz *FOREACH* vytváří smyčku pro parsování a nalezení všech záznamů. Každý ze standardních RSS tagů je snadno řešen a vložen do HTML kódu, což umožňuje výstup dat do šablony. Více informací viz. Šablony emailů.

Produktové datové zdroje

Datové zdroje je možné využít i pro přenos produktové nabídky do Mailkitu a následné použití produktových informací v kampaních. Právě zde se pak projevuje síla datových zdrojů a programovatelných šablon, která umožňuje kombinovat data z vícero zdrojů a zcela automatizovaně tak personalizovat obsah na míru jednotlivých příjemců.

Pro produktové informace je možné použít kterýkoliv z běžných formátů produktových feedu pro srovnávače Heureka, Zbozi, Google a další, nebo generovat vlastní feed s potřebnými informacemi. Protože produktové feedy jsou velice rozsáhlé a je důležitá rychlost práce s daty v nich obsaženými, tyto datové zdroje se přenášejí přímo do SQL databáze a i tak je s nimi možné pracovat. Pro nastavení produktového datového zdroje se obraťte na zákaznickou podporu, která vám pomůže s jeho implementací a následným využitím.

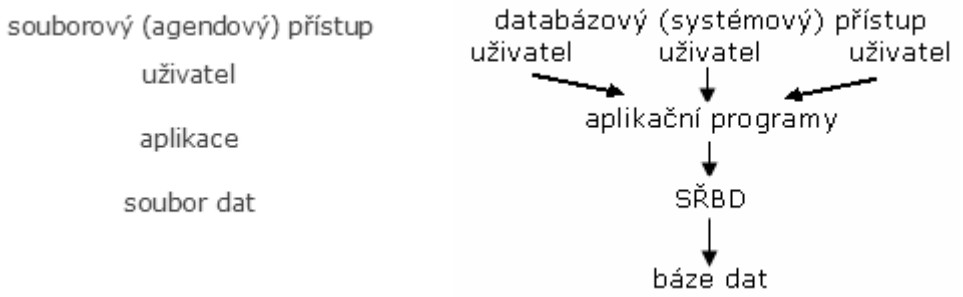
Delivery feedy

Delivery feedy jsou speciální datové zdroje, jež slouží k předání strukturovaných informací pro realizaci rozesílky kampaně. Zatím co obvykle kampaň využívá stanovený seznam příjemců, na který probíhá její rozesílka dle nastavených pravidel, v případě delivery feedu je kampaň rozesílána pouze na adresy uvedené ve feedu. Jedná se o alternativu k API volání mailkit.sendmail_mass, tzn. způsob jak do Mailkitu dostávat vysoce strukturovaná data, např. z personalizačních systémů či CRM, jež se mají při odesílání kampaně zpracovat. Tyto feedy pak musí mít striktně definovanou strukturu ve formátu XML.

9. Databázový přístup

Požadavky na databázový systém:

- kontrola konzistence dat - databáze má být schopna zajistit dodržování určitých pravidel tzv. integritních omezení a zabezpečit data před případnými nehodami, které mohou vzniknout v průběhu transakcí,
- transakce - posloupnost manipulací s daty, která musí proběhnout, aby data byla uložena správně, např. převod z 1 účtu na jiný účet v bance (musí proběhnout korektně na obou účtech),
- velké objemy dat - relativně k možnostem paměťových médií musí být databáze schopna uchovávat odpovídající objem dat,
- správa dat - etapy vývoje.



Databázový přístup

- velké databázové systémy - firmy Informix, Sybase – nákladné menší (cenově dostupnější) databázové systémy - MS Access, Paradox, FoxPro, dále pak malé databázové systémy dostupné zcela zdarma - např. My SQL
- jazyk SQL - standard umožňující využívání datových zdrojů spravovaných různými databázovými systémy

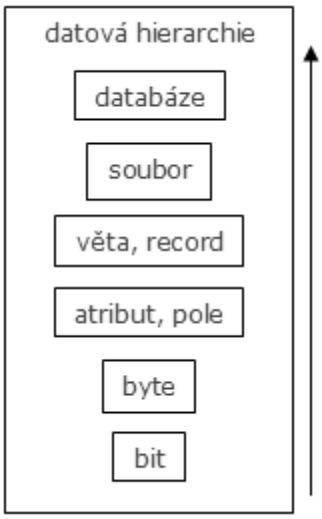
Tvorba datové základny pro IS organizace - složitá záležitost vyžadující péči lidí s různým odborným zaměřením. Při návrhu konceptuálního schématu datové základny se rozhoduje o tom, co v datové základně bude. Během provozu IS je pro uživatele důležité, zda umí datovou základnu využívat jako informační zdroj.

Požadavky na počítačový IS:

- horizontální a vertikální integrace informací,
- rychlé agregování informací od nižších stupňů zařízení,
- racionální prezentace informací v čase, formě a prostoru,
- časová frekvence,
- rozsah uchovávaných informací.

Organizace dat - soubory a databáze:

- zpracování transakcí - dávkové zpracování nebo zpracování on-line,
- současný trend - objektově orientované a hypermediální databáze.



Návrh strukturované datová základny

- realita, jejímž odrazem má být navrhovaná datová základna, sestává z různých objektů neboli entit,
- mezi sledovanými entitami mohou existovat různé vztahy (např. mezi entitami stejného typu = rekurzivní vztah).

Kardinalita vztahu: symbolické označení 1:1 (profesor XY má za manželku ZN); 1:N (které studenty učí profesor XY); M:N (které studenty učí kteří profesoři).

Integritní omezení datové základny - veškerá pravidla vymezující přípustné hodnoty (a kombinace hodnot) atributů, formát zobrazení.

Relační model dat

- Předpokládá existenci jedno hodnotových atributů:
- představa zobrazení formou relační tabulky, ve které odpovídá pojmu n-tice řádek a pojmu atribut sloupec

Relační databáze

- všechna data mají tvar 1 nebo více tabulek s pojmenovanými sloupci
- každý sloupec obsahuje data z 1 domény (tj. 1 datového typu)
- prvky jednotlivých sloupců (jímž je dáno jméno a typy) se nazývají obvykle položky nebo pole a pojem **řádek** splývá s pojmem **záznam (věta)**
- relacemi ve smyslu relačního modelu dat se obecně popisují jak entity, tak vztahy mezi nimi
- podle tohoto dvojího využití pak můžeme terminologicky rozlišovat mezi tzv. **entitními relacemi** neboli množinami uspořádaných n-tic atributů popisujících samotné entity a „vztahovými relacemi“ (tj. množiny uspořádaných n-tic)

Datové sklady

Princip tzv. Warehouse - 2 hlavní záměry:

- Sjednocení pohledu na data v jednotlivých tzv. produkčních systémech poskytne přehledný přístup k datům
- rozličně nazývané jedny a tytéž věci budou viděny jakožto věc jedna a různě měřené veličiny

10. Data strukturovaná a nestrukturovaná

10.1. Strukturovaná data

Základní typy (dělení z důvodu rozlišení povolených a nepovolených manipulací a hodnot):

- textová (řetězce znaků, vyjádření informací pomocí text. kódu, pouze určitá množina prvků, které můžeme zaznamenávat, mohou definovat syntaxi,
- číselná - čísla reálná racionální,
- datum, čas - omezeno jakých hodnot bude nabývat (30. únor, 27. hodin),
- logická - splnění podmínek existence či neexistence vlastností objektů - 2 hodnoty (0 a 1, A a N),
- kategorie - hodnota vlastností vybraná ze škály (často číselníky, umožňuje zaznamenávání hodnoty pouze kódem).
- Strukturováním je vytvořena taková organizace dat, která umožňuje efektivně uložit, zpracovat a vyhledat údaje podle potřeby → strukturovaná data vytvářejí vyhledávací klíče (někdy též identifikační klíče)- klíče, jež jednoznačně identifikují datový záznam, jsou nazývány privátní klíče (identifikační klíče) - základní podmínka a datového a databázového systému.
- Údaje o něčem (odraz reality) - jméno příjmení, adresa, věk, tel, číslo, váha, cena, ..., počet bodů, kategorie, prům. známka, ..., počet kusů, počet stran.
- Operace aneb co s nimi mohu dělat - sčítání, zaokrouhlení, násobení, připojení (jméno+příjmení), zkrácení, řazení, ..., den v týdnu, negace, ...
- Datový typ (musí být definován) - číslo, textový údaj, datum a čas, logický údaj (ano/ne).
- Zakódovaná data - různá kódování - text písmena - různé kódové tabulky (ascii, ebdic, ...) národní abecedy; datum a čas (jak píšeme datum).

10.2. Data nestrukturovaná

Data typu: volný text, audio, video, grafika, multimédia

- Poskytují více dat než pouhé strukturované údaje
- Problém: podle nestrukturovaných dat lze velmi těžko vyhledávat - používáme řešení - nestrukturovaná data bývají doplněna daty strukturovanými (název mp3)

10.3. Objemy dat - strukturovaná i nestrukturovaná data

- stránka textu ascii (notepad) 1,8 kB
- stránka textu word 50 kB
- vektorová grafika A4 30 kB
- bitmapový obraz A4(jpg, rgb) 5 MB
- záznam 1 minuty zvuku (wav) 10 MB
- záznam 90 minut obrazu 3 GB

Datový sklad (anglicky Data Warehouse, případně DWH) je zvláštní typ relační databáze, která umožňuje řešit úlohy zaměřené převážně na analytické dotazování nad rozsáhlými soubory dat.

Orientace na subjekt

U běžné relační databáze je obvyklá snaha o co nejmenší redundanci uložení dat, které je dosahováno jejich normalizací do třetí normální formy a vnitřním provázáním jednotlivých logických funkčních celků. V datovém skladu je naproti tomu řešení vždy vedeno snahou o jasnou vnitřní separaci jednotlivých funkčních celků – výsledkem je struktura, která je čitelnější pro uživatele (manažera, business analytika) za cenu zvýšených nároků na paměťový prostor.

Integrovanost

Běžná provozní aplikace (program) nad relační databází řeší určitý specifický okruh úloh nad „svými“ specifickými daty. V datovém skladu je třeba naproti tomu shromáždit informace z mnoha různých zdrojů a seskupit je nikoliv podle původu, ale podle logického významu (úzce souvisí s orientací na subjekt – všechna data týkající se určité funkční oblasti potřebuji mít „na jedné hromadě“ bez ohledu na to, odkud pocházejí).

Nízká proměnlivost

Data jsou do datového skladu obvykle nahrávána ve větších dávkách (například v denních nebo týdenních intervalech) a pak již nejsou nijak modifikována.

Historizace

Data jsou v datovém skladu obvykle udržována v historické podobě, nikoliv pouze v aktuálním stavu. To je dáno nutností provádění analýz zaměřených na vývoj v čase. V běžné relační databázi je z pohledu uživatelů obvykle zajímavý pouze aktuální stav datových objektů.

10.4. Technologické charakteristiky datového skladu

Z požadavků na datový sklad vyplývají jeho technologické charakteristiky:

- Datový sklad musí obsahovat nástroj pro nahrávání dat z různých datových zdrojů, tyto zdroje mohou mít různé datové formáty a různé fyzické umístění,

nemusí se zdaleka jednat pouze o relační databáze.

- Datový sklad ukládá data nikoliv s ohledem na co nejlepší podmínky pro editaci, ale s ohledem na co nejlepší a nejrychlejší provádění složitých dotazů – proto je pro uložení dat používána často technologie OLAP.
- Nelze předem vědět, jaké dotazy a jaké úlohy budou chtít uživatelé nad datovým skladem v budoucnosti řešit. (V době budování datového skladu je obvykle známý pouze typ úloh, nikoliv všechny jednotlivé dotazy a úlohy.) Z toho vyplývá potřeba dostatečně flexibilních a přitom uživatelsky přívětivých analytických nástrojů.

10.5. Logická struktura datového skladu

Data v datovém skladu jsou z logického (uživatelského) pohledu členěna do schémat – každé schéma odpovídá jedné analyzované funkční oblasti.

Jádro každého schématu tvoří jedna nebo několik faktových tabulek. V nich jsou uložena vlastní analyzovaná data – číselné a finanční hodnoty, které jsou použity k analytickým výpočtům – agregacím, třídění apod. Většinu paměťového místa v datovém skladu zabírají faktové tabulky, které obsahují detailní údaje ze všech zdrojů – tedy řádově více údajů než ostatní tabulky.

Faktové tabulky jsou pomocí cizích klíčů spojeny s dimenzemi. Dimenze jsou tabulky, které obsahují seznamy hodnot sloužících ke kategorizaci a třídění dat ve faktových tabulkách.

Příklad

V datovém skladu je třeba uložit informace o všech prodejích z pokladen hypermarketů, data budou dále analyzována na základě doby prodeje, prodejny, typu zboží, dodavatele, probíhajících marketingových akcí a způsobu platby (kartou, hotově).

Schéma Prodej bude obsahovat faktovou tabulku Položky prodeje, kde bude pro každou prodanou položku uložen údaj o typu prodaného zboží, ceně a počtu kusů (případně prodané hmotnosti).

Kromě této faktové tabulky bude schéma obsahovat také dimenze pro třídění položek prodeje: časové dimenze Datum a Hodina (v rámci dne), dimenzi Prodejna, dimenzi Typ zboží, kde bude jeden řádek pro každou jednotlivou položku (například „Choceňský jogurt borůvkový 250ml“), dimenzi Kategorie zboží (obsahující řádky jako například „Jogurt“), dimenzi Oddělení (obsahující řádky jako například „Mléčné výrobky“), dimenzi Dodavatel (obsahující řádky jako například „Choceňská mlékárna a.s.“) a tak dále.

Faktová tabulka musí být pomocí cizího klíče přímo nebo nepřímo propojena s každou z těchto dimenzí.

V uložení hierarchických dimenzí mám v zásadě dvě možnosti:

- Z celé hierarchie vytvořím jednu dimenzní tabulku, ve které budou údaje pro vyšší stupně hierarchie uloženy redundantně. Vznikne schema, kde je každá dimenzní tabulka vázána přímo na faktovou tabulku – podle tvaru svého diagramu se takové schéma nazývá hvězda (Star schema).
- Na hierarchickou dimenzi budu aplikovat normalizační doporučení 3NF, takže pouze dimenze na nejnižším stupni hierarchie bude vázána přímo na faktovou tabulku, ostatní pak na některou z nižších dimenzí v hierarchické struktuře – podle tvaru svého diagramu se takové schéma nazývá vločka (Snowflake schema).

11. Technologie AJAX

AJAX (Asynchronous JavaScript and XML) je v informatice obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich kompletního znovunačítání za pomoci asynchronního zpracování webových stránek pomocí knihovny napsané v JavaScriptu. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů.

Tyto aplikace jsou vyvíjeny s využitím technologií:

- HTML (nebo XHTML) a CSS pro prezentaci informací;
- DOM a JavaScript pro zobrazování a dynamické změny prezentovaných informací;
- XMLHttpRequest pro asynchronní výměnu dat s webovým serverem (typicky je užíván formát XML, ale je možné použít libovolný jiný formát včetně HTML, prostého textu, JSONči EBML).
- Podobně jako DHTML, LAMP nebo SPA, Ajax ve skutečnosti není konkrétní jednotlivá technologie, ale pojem označující použití několika technologií dohromady s určitým cílem.

Výhody

Mezi výhody patří odstranění nutnosti znovunačtení a překreslení celé stránky při každé operaci, které jsou nutné u klasického modelu WWW stránek. Pokud například uživatel klikne na tlačítko pro udělení hlasu v nějaké anketě, celá stránka se musí znovu načíst ze serveru, třebaže se na ní jen například aktualizují výsledky hlasování a veškerý zbytek obsahu zůstává stejný. Prostřednictvím AJAXu proběhne odeslání hlasu uživatele na pozadí, server zašle jen ty části stránky, které se změnily, a jen tyto části se uživateli na stránce aktualizují a překreslí. Taktéž nedochází k nepříjemnému efektu, kdy se po dané akci v průběžně načítané stránce postupně přizpůsobují a „za běhu“ formátují a zarovnávají její blokové elementy, obrázky atd. – obtěžující může být i to, že po dané akci uprostřed delší stránky (odscrollované dolů) se nově načtená stránka zobrazí vyscrollovaná nahoru. S AJAXem má uživatel pocit mnohem větší plynulosti práce, která se (zejména u rychlejšího internetového připojení) blíží běžným desktopovým aplikacím.

Z toho vyplývá také potenciál snížit zátěž na webové servery a síť obecně. Jelikož není potřeba při každém požadavku sestavit celý HTML dokument, ale pouze provedené změny, je množství vyměňovaných dat výrazně nižší a teoreticky to může mít příznivý vliv i na zátěž databázových serverů či dalších backendových systémů.

Nevýhody

AJAX však naopak může zvýšit *počet* vyměňovaných HTTP požadavků, a třebaže přenášejí nižší množství dat, tak při nevhodné implementaci zátěž neklesne.

Mezi nevýhody patří hlavně změny v paradigmatu používání webu: webové stránky se chovají jako plnohodnotné aplikace se složitou vnitřní logikou, nikoli jako posloupnost stránek, mezi kterými se lze navigovat i pomocí tlačítek Zpět a Další. Obdobným způsobem není možno předat URL stránky, ve které již bylo pomocí technologie AJAX něco „naklikáno". Moderní AJAXové aplikace jsou schopny procházení v historii (přinejmenším částečně) obnovit za použití různých technik (např. využití části adresy za znakem # či pomocí neviditelných IFRAMEs). To ale ve výsledku ztěžuje návrh stránek a vyžaduje více času a práce na implementaci pomocí AJAXu.

Problémem AJAXových aplikací také může být síťová latence: potřeba komunikace přes Internet má negativní dopady na rychlost odezvy a interaktivitu uživatelského rozhraní. Pokud uživateli není jasně signalizováno, že aplikace zpracovává jeho požadavek (a na pozadí komunikuje se serverem), jediné, co zaregistruje, je zpožděná reakce (mezitím se dokonce může snažit operaci spustit znovu, neboť se domnívá, že systém jeho příkaz ignoroval, a tím vygenerovat větší zátěž serveru a/nebo způsobit něco, co neměl v úmyslu, např. objednat deset vstupenek místo jedné). Jako vhodné řešení se doporučuje po dobu mezi odbavením požadavku na server a jeho odpovědí nějakým způsobem zobrazit, že uživatelův požadavek se zpracovává, například textem nebo animovanou ikonou.

Další nevýhodou AJAXu je nutnost používat moderní grafické prohlížeče, které podporují potřebné technologie. Všechny dnešní běžné prohlížeče však tyto technologie alespoň v základu podporují, problém tak zůstává jen u minoritních prohlížečů typu Lynx nebo na hardwarově slabších zařízeních pro prohlížení, například na (některých) mobilních telefonech a PDA. V rámci webové přístupnosti se vyžaduje, aby stránky byly přístupné i z prohlížečů bez podpory AJAXu, což pro vývojáře znamená více času a práce a objednavatelům výsledných stránek větší náklady.

AJAX

Zkratka AJAX pochází z anglického Asynchronous JavaScript and XML. AJAX je moderní technologie často využívaná v současných webových aplikacích. Hodně se o ní mluví, neboť je součástí RIA, nového směru programování vedoucímu k vyššímu uživatelskému komfortu a funkčnosti aplikací.

AJAX však ve skutečnosti není žádnou novou technologií, pouze novou kombinací technologií již dávno známých, tj. HTML (nebo XHTML), JavaScriptu, XML a XMLHttpRequest.

A proč je AJAX tak výhodný? Aplikace využívající AJAX umí odeslat a získat zpět data ze serveru bez nutnosti znovu nahrávat celou stránku (na rozdíl od klasických odkazů). AJAX tak může mít mnohá využití, příkladem mohou být různé našeptávače (formuláře, které se automaticky předvyplňují podle stisknuté klávesy), AJAX ankety a další složitější aplikace, které dokáží uživateli usnadnit práci.

AJAX však má i určité nevýhody, především při nevhodném užití snižuje významně

použitelnost stránek. Proto je třeba, stejně jako u jiných technologií, AJAX aplikaci dobře promyslet a před nasazením i důkladně otestovat na uživatelích.

Jak jistě většina z Vás ví, Javascript je tzv. client-side jazyk. Provádí se tedy na straně webového prohlížeče na klientském počítači a je interpretován javascriptovým interpreterem. Disponuje řadou vlastností, mimo jiné i tou, která nás teď bude zajímat nejvíce – schopností provádět asynchronní operace/úlohy.

V čem vlastně spočívá asynchronnost v AJAXu? Jde o schopnost Javascriptu zavolat na serveru nějaký skript, nebo prvek API a nečekat bezprostředně na odpověď. Místo toho pokračuje provádění kódu (uživateli se například zobrazí stránka a může s ní běžně pracovat). Když potom odpověď přijde, provádění hlavního kontextu kódu se pozastaví (dříve, či později v závislosti na prioritě prováděné úlohy) a dojde k zavolání tzv. callbacku – funkce, která se provede v případě, že ze serveru dostaneme odpověď. Takovémuto kódu, či takovéto funkci se pak říká neblokující, protože neblokuje průběh provádění skriptu čekáním na odpověď, či zpracování.

Co tedy AJAX je a co nám přináší? Jde o zkratku Asynchronous Javascript And XML, to nám zatím neřekne víc, než že to souvisí s XML a Javascriptem, nicméně půjdeme na to postupně a všechno si vysvětlíme.

AJAX je způsob, jak programovat v Javascriptu – není to tedy žádný nový jazyk ani framework nebo knihovna třetí strany. Jde o způsob, kterým můžeme vyměňovat data v naší aplikaci s databází, serverovými skripty (PHP, Java, ASP, atd.) aniž bychom aktualizovali/reloadovali celou stránku, technicky vzato neaktualizujeme (ve smyslu refreshe) vůbec žádnou část naší aplikace, či stránky a v tom vlastně tkví ta "magická síla" a podstata AJAXu.

Proč se v názvu vyskytuje zkratka XML? Jednoduše je to jeden z formátů dat, ve kterém AJAX umí obdržet informaci ze serveru, nejznámější a často používané jsou JSON, XML, text, binární data. Každý z nich má něco do sebe, každý se hodí na něco jiného *a používá se trochu jinak.

AJAX vám například dává možnost zkontrolovat data zadávaná uživatelem, ještě než je uživatel odešle potvrzením formuláře. Do jisté míry je toho schopný i doteď známý Javascript, ale co když jde o registrační formulář s podmíněným heslem, uživatelským jménem nebo e-mailem, které musí být unikátní v rámci databáze/tabulky? Přesně pro tyto a mnohé další účely je tato technologie to pravé. Dovolí vám totiž provést ty samé operace, které byste prováděli voláním například PHP skriptů při přesměrování po odeslání formuláře, ale na rozdíl od nich vám umožní provádět tato volání na pozadí.

Ajax v základech

Myšlenka Ajaxu spočívá v tom, že jsou části webové stránky natahovány asynchronně a tímto způsobem se mění obsah stránky. Znamená to, že při změně obsahu nebude znovu natažena celá stránka, nýbrž pouze modifikovaný obsah. Za tímto účelem je z JavaScriptu volán objekt XMLHttpRequest. Tento požadavek (Request) se dotazuje serveru na obsah, který je typicky vracen zpět pomocí XML a JavaScriptu. Skript pak toto XML analyzuje. Data z XML skript potom překládá a odpovídajícím způsobem mění pomocí

DOM (Document Object Model) webovou stránku.

Základní principy práce s Ajaxem si popíšeme na příkladu, který má blízko k jeho praktickému využití – konkrétně se jedná o diskusní fórum, v němž jsou diskuse v HTML zobrazeny jako strom. Aby si uživatel mohl přečíst daný příspěvek, musí kliknout na jeho nadpis. V případě klasického modelu by byl zaslán požadavek na server a následně je vytvořena celá stránka, která kromě původního stromu obsahuje navíc i text příspěvku. U modelu založeného na Ajaxu bude jako reakce na dotaz na server poslán pouze samotný text příspěvku: Připojení tohoto textu k jinak nezměněné stránce zajistí příslušný program v JavaScriptu.

Pro provedení této úlohy byl u starších verzí Internet Exploreru (počínaje verzí 5.0) k dispozici ActiveX objekt. IE 7 už stejně jako jiné prohlížeče dává odpovídající objekt k dispozici nativně. Aktuální API XMLHttpRequest Internet Exploreru je k dispozici na webu MSDN a dostupné je rovněž API projektu Mozilla.

Vlastním trikem Ajaxu je právě XMLHttpRequest – ten může být totiž zpracováván asynchronně, což v této souvislosti znamená asynchronně k uspořádání zbytku stránky a rovněž asynchronně vůči na stránce běžícím skriptům. To znamená, že stažení dat může být provedeno prostřednictvím HTTP a že data mohou být stažena bez toho, aby tím trpěla interakce uživatele se stránkou. Sestavení požadované stránky běží také na pozadí.

Co to vlastně je AJAX? Stručně řečeno, AJAX je technologie, která pomocí skriptů zprostředkovává stránce komunikaci s webovým serverem. Lze tak měnit obsah stránky bez potřeby ji znovu načítat.

Teoreticky lze tedy vytvořit dynamické stránky bez serverových skriptů, tedy mít jednu stránku pro celý web, která pouze mění svůj obsah. Toto řešení však nedoporučuji, protože, jak již bylo řečeno, se jedná o scriptovou technologii, tudíž je příliš závislá na klientu a není zaručeno, zda bude fungovat. Využití však najde v doplňcích webové stránky, jako je anketa, nebo pomocník pro vyhledávání, jako má např. Seznam.cz. Zde je poměrně jednoduché udělat alternativní řešení - u ankety se hlas pošle klasickým formulářem a u vyhledávání zas chybějící pomocník nikoho příliš trápit nebude.

Pokud máte jistotu, že u návštěvníků vašich stránek AJAX poběží, je možné jej použít i pro složitější aplikace. Osobně například používám chat, který si stahuje pouze nové příspěvky a pokud je na něm návštěvník sám, kontrola aktualizací se zpomalí, což šetří mnoho přenesených dat.

12. Frameworky

Framework (aplikační rámec) je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, podporu pro návrhové vzory nebo doporučené postupy při vývoji.

- Účel
- Architektura
- Příklady
- Související články
- Externí odkazy

Účel

Cílem frameworku je převzetí typických problémů dané oblasti, čímž se usnadní vývoj tak, aby se návrháři a vývojáři mohli soustředit pouze na své zadání. Například tým, který používá Apache Struts k vývoji webových stránek pro banku, se může zaměřit na to, jak se budou provádět bankovní operace, a ne jak zajistit bezchybnou navigaci mezi jednotlivými stránkami.

Vyskytují se námitky, že použitím frameworku bude kód pomalý či jinak neefektivní a že čas, který se ušetří použitím cizího kódu, se musí věnovat nastudování frameworku. Nicméně při jeho opakovaném nasazení nebo ve velkém projektu dojde k výrazné úspoře času. Při odinstalování frameworku již nebude možné některé aplikace spustit.

Architektura

Framework se skládá z tzv. frozen spots a hot spots. Frozen spots definují celkovou architekturu softwarové struktury, její základní komponenty a vztahy mezi nimi. Tyto části se nemění při žádném použití frameworku. Naproti tomu hot spots jsou komponenty, které spolu s kódem programátora vytvářejí zcela specifickou funkcionalitu, a proto jsou skoro pokaždé jiné.

V objektově orientovaném prostředí je framework tvořen abstraktními a klasickými (neabstraktními) třídami. Frozen spots pak mohou být reprezentovány abstraktními třídami a vlastní kód (hot spots) se přidá implementací abstraktních metod.

Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovnu API, návrhové vzory nebo doporučené postupy při vývoji.

Další velmi důležitým faktorem při výběru je to, k čemu framework potřebuji a k čemu je určen. Obecně lze frameworky rozdělit na dvě skupiny:

- Sady scriptů – knihoven – pokrývající všemožné potřeby
- Scripty vytvářející jednu konkrétní webovou aplikaci

Výhody

- Rychlejší vývoj
- Méně kódu
- Univerzální kód – změny či nové funkce přidáte 3× jednodušeji
- Pěkná URL

Nepochybně se vaše rychlost vývoje zrychlí. Framework vám usnadní práci. Už nebudete programovat rutiny jako je připojení k databázi, zacheckování správného option u selectu, nebo zvalidování formuláře. Cool-Url budou pro vás automatická věc. Už žádné vrásky s XSS. Oddělená aplikační a prezentační logika, super! Změna DB serveru z MySQL na PostgreSQL? Ne, to vás nemůže rozhodit.

Díky frameworku se můžete opravdu soustředit jen na ten opravdový vývoj. Neřešíte blbůstky. To vše je ale vykoupeno časem, kdy se učíte s frameworkem pracovat. Znáte to, něco konečně uděláte, ale za týden, až proniknete ještě hlouběji do dané problematiky, zjistíte, že to šlo třikrát jednodušeji. Nelze říct, že učením frameworku strávíte měsíc. Ne, někdy, u těch složitějších a komplexnějších, to může být i rok. Nakonec si začnete framework rozšiřovat o své knihovny. A za rok se ohlédnete a zjistíte, že to, co byste předtím programovali týden, máte už teď za jeden den. Framework je úspora času a čas, to jsou peníze.

Výběr není jednoduchý. Jsem s to, že je třeba mít aspoň dva ve své nabídce. Jeden na věci složitější, druhý na ty jednoduší. A pak před každým projektem učinit rozhodnutí, co je pro něj vhodnější. Samozřejmě, nejlépe se naučit ty nejznámější, ty, které mají dobrou velkou komunitu, ty, u kterých máte jistotu, že vývoj za týden neskončí.

- CakePHP: celkem jednoduchý na naučení; celkem mocný; kvalitní komunita; dlouhý vývoj nové verze
- Zend Framework: velmi mocný; pokrývající veškeré potřeby při tvorbě jakýkoliv webových aplikací; veliká komunita; složitý; dlouhé názvy;
- CodeIgniter: maličký, jednoduchý; vyvíjen jednou firmou, ne komunitou; v něčem lepší než Cake, v něčem velmi „hloupý“ – např. nemá layouty

Monolitické frameworky se postupně rozpadají do samostatných (decoupled) komponent. A to přináší řadu výhod. Zatímco dříve bylo použití jen jedné části frameworku obtížné až nemožné, dnes si prostě nainstalujete jeho komponentu. Vývojový cyklus jednotlivých komponent může mít různé tempo. Mají vlastní repozitáře, issue trackery, mohou mít vlastní vývojářské týmy.

Komponenty můžete aktualizovat na nové verze průběžně, bez čekání, než vyjde další verze celého frameworku. Nebo naopak se můžete rozhodnout určitou komponentu neaktualizovat, třeba kvůli BC breaku.

Význam slova framework se tak posouvá, o verzích už takřka nelze hovořit. Místo frameworku XYZ ve verzi 2.3.1 používáte sadu komponent v různých verzích, které spolu fungují.

12.1. Co jsou frameworky?

Při tvorbě moderních webových aplikací se poměrně často využívají tzv. frameworky, neboli vývojové rámce. Jedná se o soubor knihoven a zdrojového kódu, který lze opakovaně použít k usnadnění práce a pokrýt jeho funkcionalitou část tvořené aplikace. Frameworky dokáží řešit problémy, které lze nějakým způsobem zobecnit, jelikož by jinak nemělo smysl tvořit framework. Tvůrce aplikace se tak může více koncentrovat na funkce pro aplikaci výjimečné a jedinečné a neřešit problémy týkající se rutinních úkolů.

Využití javascriptových frameworků

Framework lze vytvořit ke většině programovacích jazyků. Proto javascriptový framework je framework, který slouží k ulehčení práce a programování v javascriptu. Jejich využití je značně široké a každý dnem se rozšiřuje portfolio problémů, které javasriptové frameworky dokáží řešit. Obecně lze říci, že je lze využít ke psaní efektivního zdrojového kódu, z velké části řeší také vzájemnou kompatibilitu (resp. nekompatibilitu) mezi webovými prohlížeči a napsaného kódu, šetří programátorův čas a umožňují využít prvků, které by bylo jinak velmi obtížné programovat svépomocí. Jejich síla a využití je spojena především s technologií AJAX, kde vylepšují vzájemnou interakci uživatele a aplikace pomocí asynchronní komunikace mezi klientem a serverem. Dále pak umožňují dynamicky a hlavně jednoduše přistupovat a měnit jednotlivé elementy stránky (DOM model), GUI prvky, přiřazovat jim události a efektně je animovat. V neposlední řadě některé frameworky obsahují již předpřipravené GUI komponenty, které buď není možno pomocí jiné technologie realizovat, nebo je to příliš náročné a neefektivní. S těmito prvky je možno jednoduše v rámci frameworku pracovat a začlenit je do aplikace jen pomocí několika řádků kódu.

13.10 nejlepších PH frameworků pro vývojáře

PHP, známý po celém světě jako nejoblíbenější skriptovací jazyk na straně serveru, se hodně rozvinul od dob, kdy se ve statických HTML souborech začaly objevovat první inline fragmenty kódu.

V těch časech museli vývojáři budovat složité weby a webové aplikace, a nad jistou úrovní složitosti **zabíralo příliš mnoho času a úsilí vždycky začínat úplně od začátku**. Odtud vznikla potřeba strukturovanějšího přirozeného způsobu vývoje. Adekvátní řešení této potřeby poskytují vývojářům PHP frameworky.

Proč vůbec používat nějaký PHP framework

Podívejme se nejprve na nejpádnější důvody, proč mnozí vývojáři rádi používají PHP frameworky, a jak mohou tyto frameworky pozvednout úroveň vývojového procesu. Co PHP frameworky poskytují:

- Umožňují rychlý vývoj.
- Poskytují dobře organizovaný, opětovně využitelný a udržovatelný kód
- Umožňují růst v průběhu času, protože webové aplikace běžící na frameworcích jsou škálovatelné.
- Zbavíte se starostí souvisejících s nízkoúrovňovou bezpečností webu.
- Dodržují vzor MVC (Model-View-Controller), který zajišťuje separaci prezentace a logiky.
- Prosazují moderní webové vývojové postupy, mezi něž patří nástroje objektově orientovaného programování.

1. Laravel

Přestože je Laravel poměrně nový PHP framework (byl vydán v roce 2011), je podle nejnovějšího online průzkumu na webu Sitepoint nejoblíbenějším frameworkem mezi vývojáři. Laravel má obrovitý ekosystém s platformou připravenou k okamžitému hostování a rozmisťování a jeho oficiální web nabízí mnoho návodů ve formě screencastů, jimž zde říkají Laracasty.

2. Symfony

Komponenty frameworku Symfony 2 používají mnohé impozantní projekty, jako jsou systém pro řízení obsahu Drupalnebo software phpBB pro spouštění fór, ale **spoléhá se na něj i** Laravel – framework uvedený výše. Symfony má **rozsáhlou vývojářskou komunitu** a mnoho vášnivých příznivců.

3. CodeIgniter

CodeIgniter je odlehčený PHP framework, který už je téměř deset let starý (původně byl vydaný v roce 2006). CodeIgniter má velmi přímočarý instalační proces, který požaduje jen minimum konfigurace, takže může ušetřit soustu trápení. Je také ideální volbou, pokud se chcete **vyvarovat konfliktů s verzemi PHP**, protože **funguje hladce téměř na všech sdílených a dedikovaných hostovacích platformách** (v současné době požaduje jen PHP 5.2.4).

4. Yii 2

Pokud zvolíte framework Yii, dodáte svému webu vzpruhu pro výkon, protože je **rychlejší než ostatní PHP frameworky**, používá totiž **extenzívně techniku „pohodového" načítání (lazy loading).** Yii 2 je čistě **objektově orientovaný** a je založený na kódovacím pojetí **DRY** (Don't Repeat Yourself, „neopakujte se"), takže poskytuje **dost jasnou a logickou kódovou bázi**.

5. Phalcon

Framework Phalcon byl vydán v roce 2012 a rychle si mezi vývojáři PHP vydobyl popularitu. Říká se o něm, že je rychlý jako sokol (anglicky falcon), protože **byl napsán v jazycích C a C++, aby se dosáhlo nejvyšší možné úrovně optimalizace výkonu**. Dobrou zprávou je, že se nemusíte učit jazyk C, protože funkcionalita je **vystavena jako PHP třídy připravené k okamžitému použití v jakékoli aplikaci**.

6. CakePHP

CakePHP už má za sebou celou dekádu existence (první verze byla vydaná v roce 2005), pořád však patří mezi nejoblíbenější PHP frameworky, protože vždy dbal na to, aby držel krok s dobou. Nejnovější verze CakePHP 3.0 rozšířila správu relací, **vylepšila modularitu** tím, že oddělila několik komponent, a zvýšila způsobilost **vytvářet soběstačnější knihovny**.

7. Zend Framework

Zend je robustní a stabilní PHP framework zabalený spolu se spoustou konfiguračních voleb, proto se obvykle **nedoporučuje pro menší projekty,** je však **vynikající pro ty složitější**. Mezi partnery Zend patří takové společnosti, jako IBM, Microsoft, Google a Adobe. Nadcházející hlavní vydání, Zend Framework 3, bude optimalizované pro PHP 7, bude však i nadále podporovat PHP 5.5.

8. Slim

Slim je PHP mikro framework, který poskytuje vše, co potřebujete, a nic, co nepotřebujete. Mikro frameworky jsou v designu minimalistické, jsou **vynikající pro menší aplikace**, pro které by byl framework, který má plnohodnotnou výbavu pro úplně všechno, něco jako kanón na vrabce. Tvůrce Slimu se inspiroval mikro frameworkem Ruby, který se jmenuje Sinatra.

9. FuelPHP

FuelPHP je flexibilní plnohodnotný PHP framework, který podporuje nejen obyčejný vzor

MVC, ale na úrovni architektury též jeho rozvinutou verzi, HMVC (Hierarchical Model-View-Controller). FuelPHP přidává mezi vrstvy Controller a View **nepovinnou třídu** s názvem Presenter (dříve se jmenovala ViewModel), aby **obsahovala logiku potřebnou ke generování pohledů**.

10. PHPixie

PHPixie je zcela nový framework, odstartoval v roce 2012 s cílem vytvořit vysoce výkonný framework pro weby určené jen ke čtení. PHPixie stejně jako FuelPHP **implementuje návrhový vzor HMVC** a je vybudovaný pomocí nezávislých komponent, které **lze používat také bez samotného frameworku**. Komponenty PHPixie jsou stoprocentně prověřené přes jednotkové testy a požadují jen minimum závislostí.

WEBOVÉ TECHNOLOGIE

1. HTML

HTML neboli **HyperText Markup Language** je jazyk pro tvorbu webových stránek – aktuální verze – HTML5 (HTML 5.2). Tento jazyk je standardizován organizací W3C.

Skládá se z tagů (značek) a atributů (vlastností).

Zobrazení HTML probíhá pomocí prohlížeče a to v několika krocích (=parsování).

- Prohlížeč načte dokument a provede syntaktickou analýzu podle DTD
- Ke každému prvku je přiřazen styl (způsob zobrazení)
- Aplikace předepsaného kódu skriptovacích jazyků
- Postupné vykreslení stránky

HTML značky dělíme podle významu na:

- Strukturální, které určují strukturu dokumentu. Např.: Odstavec (<p>), nadpisy (<h1>, <h2>…)…
- Popisné (sémantické), popisující povahu obsahu prvku, např.: Nadpis (<title>), adresa (<address>)…
- Stylistické, které určují vzhled prvku při zobrazení, např.: Tučné písmo (<b>), kurzíva (<i>), zvýraznění ( )…

Každý HTML dokument by měl dodržovat **základní schéma**:

Na začátku je DTD direktiva, deklarace typu dokumentu:

```
<! DOCTYPE html >
```

Následuje kořenový element:

```
< html >          </ html >
```

A hlavička dokumentu (Obsahuje meta data):

```
<head>
<meta charset = "utf-8"> - Kodierung
<title > Seitenüberschrift </ title >
```

Autor, popis, klíčová slova, kaskádové styly…

```
< head >
```

Za hlavičkou následuje tělo dokumentu, které zahrnuje obsah stránky jako text, obrázky, odkazy, tabulky…

```
<body> … </body>
```

Pro psaní HTML je výhodné použít textový editor, který zvládá barevnou syntaxi (barevně rozlišuje jednotlivé části kódu – značky, vlastnosti, prostý text), dokáže napovídat značky, zná chytré tabulátory nebo zvládá validovat dokument podle předepsané specifikace.

Například: Notepad++, PSPad…

Případně lze použít WISIWYG (What You See Is What You Get) editory, které pracují přímo s hotovou stránkou. Uživatel tak nemusí znát jazyk HTML –poskládá stránku a editor vygeneruje příslušný kód.

Například: Adobe Dreamweaver, Microsoft Expression Web, LibreOffice Writer/Web, Bluegrifon.

2. CSS – Kaskádové styly

CSS je jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML, XHTML nebo XML vyvinutý a standardizovaný organizací W3C. Aktuální verze je

CSS 3.

CSS lze připojit do HTML kódu několika způsoby:

- V kódu html stránky pomocí elementu <style> </style>

```
<style = "text/css ">
# head {
    width : 200px;
    height : 450px;
}
</style>
```

- Pomocí externího souboru - element <link>

```
<head>
<link rel = 'style sheet' href = 'styles.css' type
= 'text/css '>
<head>
```

- Přímý inline zápis stylu pomocí atributu style

```
<p style = " color : red ; text- decoration : underline ">
Tento odstavec bude červený a podtržený.</ p>
```

Jazyka PHP se skládá z pravidel, kde každé pravidlo obsahuje selektor a blok deklarací. Každý blok deklarací pak obsahuje jednotlivé deklarace oddělené středníkem. Každá deklarace se skládá z identifikátoru vlastnosti, dvojtečky a hodnoty. Může ještě následovat označení !importatn, které zvyšuje sílu deklarace.

Proč používat CSS místo formátování pomocí HTML?

CSS nabízí:

- větší možnosti formátování, které nenabízí HTML– např.: určení vzdálenosti elementů od okrajů stránky.
- Jednodušší provádění úprav vzhledu – změna barvy všech nadpisů najednou, změna fontu,...
- Možnost vytvoření šablony pro několik stránek najednou
- Oddělení struktury a stylu – v HTML obsah, v CSS vzhled
- Cachování stylů – rychlejší načítání stránek (ale navíc jeden HTTP požadavek na načtení externího souboru)
- Dynamické změny CSS vlastností pomocí Javascriptu
- Pomocí CSS lze formátovat jakýkoli jazyk XML
- Nastavení zobrazení pro jednotlivá zařízení – podmíněné CSS

Koncový uživatel si může napsat svůj vlastní CSS styl pro libovolnou stránku - lze nastavit, aby všechny odkazy na všech webech byly vždy podtržené nebo aby na tomto konkrétním webu mělo písmo vždy černou barvu.

V kombinaci s Javascriptem mohou vzniknout účinné bookmarklety, které mohou různě vylepšovat vzhled stránky. Například odstranit všechny obrázky na pozadí, změnit pozadí na bílé a písmo na černé apod.

S CSS se také pojí jisté nevýhody. Tou hlavní je, ne vždy dostatečná podpora v majoritních prohlížečích, případně chyby v implementaci CSS v prohlížečích. Toto lze obejít použitím různých stylů pro různé prohlížeče.

Lze použít podmíněné komentáře pro nastavení různých prohlížečů

```
<!--[if IE]> <style type="text/css"> # alert {color: blue;} </style> <![endif]-->
```

Tento kód bude interpretován pouze Internet Explorerem, ostatní prohlížeče uvidí obyčejný HTML komentář a interní stylopis tak nebudou interpretovat.

I CSS má své určité limity.

- CSS selektory neposkytují přístup k rodičovským prvkům
    - Nemůžete například nastylovat jen ty odstavce, které obsahují odkaz.
- Horizontální kontrola prvků na stránce je intuitivní a jednoduchá, naopak vertikální stylování vyžaduje komplexnější přístup (např.: flexbox nebo grid).
- CSS neposkytuje možnost pro symbolický zápis proměnné nebo konstanty
- Všechny hodnoty musí být vepsány přímo v kódu.
    - Například pokud se na vícero místech používá stejná barva, nemůže se použít symbolický zápis barva=red; a poté už jen psát proměnnou barva, všude se musí uvádět přímo hodnota red. Toto omezení odstraňují CSS preprocesory (např. SASS, LESS, Stylus).
- CSS2 nenabízí žádnou možnost pro tvorbu kulatých rámečků nebo jiných kulatých objektů. Pracuje pouze s obdélníky.
- CSS2 nenabízí žádnou možnost, jak jednomu elementu přiřadit více obrázků na pozadí.

3. CSS II

Je potřeba si ujasnit, co která verze CSS nabízí a definuje.

CSS verze 1 zavádí syntaxi používanou i v následujících verzích, přináší způsob vybírání prvků přes selektory, několik pseudotříd a definuje hodnoty a jejich jednotky.

Kategorie, pro které CSS 1 definuje hodnoty a jednotky:

- vlastnosti písma
- barvy textu a pozadí
- vlastnosti textu
- vlastnosti blokových elementů
- způsoby zobrazení prvků
- řízení pozice

Hodnoty a jednotky:

u desetinných hodnot se používá desetinná tečka

Jednotky délky

- (žádné) – u bezrozměrných vlastností (např. line-height)
- % – procenta, jednotka relativní vůči implicitnímu rozměru, zapisována bez mezery
- pt – typografický bod, výchozí jednotka je 1/72 palce
- px – 0,75 pt
- pc – pica, 1 pc = 12 pt
- cm – centimetr
- mm – milimetr
- in – palec
- em – čtverčík, je rovna základní výšce písma
- ex – výška písmene "x", je relativní k použitému písmu

Barvy v CSS se zadávají pomocí RGB palety a to:

- buď číselně #rrggbb – dvouciferná hexadecimální hodnota (00 až ff) – 16 milionů barev
- #rgb – hexadecimálně jednociferně (0 až f) - 4 096 barev.
- jako bezpečné barvy, což je ještě užší podmnožina barev: Jde o jednociferné barvy, jejichž hodnoty vzrůstají o 3 - kombinace šesti hodnot {0, 3, 6, 9, C, F} = 216 barev.
- rgb(r,g,b) - v desítkové soustavě (0 až 255) a syntaxí jako funkce – 16 milionů barev
- rgb(r%,g%,b%) – hodnoty v rozsahu (0 až 100)

Případně lze použít předdefinované konstanty a textové názvy pojmenovaných barev.

Např. aqua (jasná modrozelená), black (černá) = #000, blue (modrá), fuchsia (anilínová červeň), gray (šedivá), green (zelená), lime (citrónově zelená), maroon (kaštanová), navy (námořnická modř), olive (olivová), purple (purpurová), red (červená), silver (stříbrná), teal (tmavá modrozelená), white (bílá) = #FFF, yellow (žlutá). V CSS verze 4.0 je pojmenováno 148 barev.

Písmo v CSS 1 má své atributy.

- font-style: normal, italic (kurzíva), oblique (šikmá antikva)
- font-size: medium; menší velikosti: xx-small, x-small, small; větší velikosti: large, x-large, xx-large, smaller, larger, velikost v procentech 100% je standardní velikost
- font-weight: tučnost: normal, bold, bolder, thicker, lze zadat číslem: 100, 200 … 900 (400=normal, 700=bold)
- font-variant: small-caps (kapitálky), normal
- font-decoration: underline (podtržené), overline (nadtržené), blink (blikající), line-through (přeškrtnuté), none (standardní)

CSS 1 také zavádí obecné druhy písma

- serif – klasické patkové písmo (např. Times New Roman)
- sans-serif – bezpatkové písmo (např. Helvetica nebo Arial)
- cursive – kurzíva
- fantasy – ozdobné písmo
- monospace – neproporcionální písmo (např. Courier)

Dále CSS 1 definuje prvek URL tak, že:

- konstrukce url(), kde se mezi závorky specifikuje adresa zdroje
- Absolutní: url(http://www.example.com/images/logo.png)
- Relativní vůči serveru: url(/images/logo.jpg)
- Relativní vůči aktuálnímu adresáři: url(images/logo.jpg)

Obsahuje-li URL čárky, mezery, uvozovky nebo konec kulaté závorky, tyto znaky se dají eskapovat pomocí zpětného lomítka.

Dále pak Selektory:

- Typový selektor: A - Všechny elementy typu A
- Selektor třídy: A.třída - Všechny elementy A s atributem class=„třída“
- ID selektor: A#ID - Všechny elementy s idetifikátorem ID
- Selektor následovníka: A B - Všechny elementy B uvnitř A

Definice CCS 2 přinesla další definice:

- outline – vnější ohraničení
- max-height, max-width, min-height, min-width – minimální a maximální šířka nebo výška elementu
- content – nastavitelný obsah elementu
- counter – nástroj pro číslování kapitol
- quotes – styl citací
- clip – ořezávání
- cursor – kurzor nad elementem
- position – možnost pozicovat element v řádku, v bloku, absolutně, relativně, ...
- top, bottom, right, left – okrajové hodnoty pro position:absolute;
- overflow – zobrazení při přetékání obsahu
- visibility – viditelnost elementů
- z-index – možnosti překrývání
- page-break, orphans, widows – typografická pravidla pro dělení stránek
- table-layout, border-collapse, border-spacing, caption-side, empty-cells – nové možnosti pro zobrazení tabulek
- direction – směr psaní textu

CSS 2 umožňuje užívání nových typů selektorů:

- Univerzální: * - Vztahuje se na úplně všechny elementy
- Selektor dítěte: A>B - bere v potaz pouze ty elementy B, které jsou vnořeny do A přímo
- Selektor sourozenců: A+B - vybere všechny elementy B, které mají stejného rodiče jako A a které po něm zároveň přímo následují

Definuje také selektory pomocí atributů:

- A[attr] - všechny elementy A, které mají nastavený atribut attr
- A[attr=hodnota] - všechny elementy A, které mají nastaveny atribut attr=“hodnota”
- A[attr~=hodnota] - všechny elementy A s atributem attr, jehož hodnotu tvoří seznam slov oddělených mezerou a právě jedno z těchto slov se shoduje s “hodnota”
- A[attr|=hodnota] - všechny elementy A, jejichž hodnota atributu attr začíná řetězcem “hodnota”, pak následuje pomlčka a další řetězec

Dále pak zavádí systém pseudoelementů a pseudotříd.

- Pseudoelementy
  - První řádek - :first-line
  - První písmeno (iniciála) - :first-letter
  - Před - :before
  - Za - :after
- Jazyková pseudotřída
  - :lang
- Rodičovská pseudotřída
  - První potomek - :first-child
- Pseudotřídy odkazů
  - Nenavštívený odkaz - :link
  - Navštívený odkaz - :visited
  - Zaměřený odkaz - :focus
  - Odkaz pod myší - :hover
  - Aktivní odkaz - :active

Aby vše správně fungovalo, je u odkazů potřeba dodržet pořadí. Každá pseudotřída má jinou prioritu.

A konečně poslední verze CSS3, která se pojí se standardem HTML5, jež je má plně využívat. Nabízí tak:

- animace – CSS3 přímo podporuje animaci elementů (jejich vlastností), doposud se animace dělaly přes DHTML, např. jQuery
- dodatečné možnosti stylování pozadí blokových elementů, včetně např. oříznutí jejich pozadí, vržených stínů nebo zakulacených okrajů
- dodatečná pravidla pro přetékání obsahu blokových elementů
- opacita – míra neprůhlednosti prvků
- další podpora stránkovaného obsahu – záložky a možnosti dělení textu
- flexibilní blokové elementy
- cílové odkazy – jak a kde se mají otevírat
- vlastnosti pro drag'n'drop
- dodatečné vlastnosti pro písma – načítání písma z externího zdroje, přizpůsobení velikosti při nízké čitelnosti, zužování/rozšiřování písma
- vlastnosti pro generovaný obsah – zkracování obsahu s možností expanze, přesunování elementů dále ve stránce
- mřížky (zatím nikde neimplementované)
- nové vlastnosti pro marquee
- automatický vícesloupcový layout
- nové vlastnosti Ruby
- vlastnosti pro čtený text
- 2D a 3D transformace
- vlastnosti spolupracující s navigací
- uživatelsky definované vlastnosti

4. JavaScript

Javascript je programovací jazyk používaný na webových stránkách, zapisuje se přímo do HTML kódu a patří mezi klientské skripty (vykonávají se na straně klienta)

Javascript je:

- interpretovaný – nemusí se kompilovat
- objektový – využívá objektů prohlížeče a zabudovaných objektů
- závislý na prohlížeči – funguje ale ve většině prohlížečů
- case sensitivní – záleží na velikosti písem v zápisu
- syntaxí podobný jazykům C, Java a podobným

Z podstaty Javascriptu vyplývají jistá omezení:

- Funguje pouze v prohlížeči.
- Uživatel může JavaScript zakázat
- Existují různé odlišné verze jazyka i prohlížečů, což vede k častým chybám.
- Neumí přistupovat k souborům (kromě cookies) ani k žádným systémovým objektům.
- Neumí žádná data uložit (kromě cookies).

Přesto je Javascript hojně využívaný a lze jej mnohdy použít jako vestavěný skriptovací jazyk u mnohých aplikací.

Lze jej nalézt:

- Většina rozšíření pro webové prohlížeče
- Některé NoSQL databáze jako je MongoDB nebo CouchDB akceptují dotazy napsané v JavaScriptu.
- Adobe – Acrobat and Adobe Reader, nástroje v Adobe Creative Suite (Photoshop, Illustrator, Dreamweaver a InDesign)
- Kancelářský balík aplikací OpenOffice umožňuje JavaScript používat jako skriptovací jazyk.
- Interaktivní zpracování signálu hudebního software Max/MSP
- Digitální software Apple Logic Pro X audio workstation umožňuje vytvořit vlastní MIDI efekty pluginy pomocí JavaScriptu.
- ECMAScript (JavaScript) byl zahrnut v normě VRML97 pro skriptování uzlů souborů VRML.
- Herní engine Unity 3D podporuje upravenou verzi JavaScriptu pro skriptování pomocí Mono.
- DX Studio (3D engine) používá implementaci JavaScriptu SpiderMonkey pro hry a simulace logiky.
- Maxwell Render poskytuje skriptovací engine ECMA standardu pro automatizaci úkolů.
- Google Apps Script v tabulkách Google a Google Sites umožňuje uživatelům vytvářet vlastní vzorce, automatizovat opakující se úlohy a také komunikovat s ostatními produkty Google jako je Gmail.
- Produkty SpinetiX používají SpiderMonkey JavaScript pro scriptování v SVG souborech.

Javascript lze také použít jako skriptovací engine:

- Technologie Active Scripting od Microsoftu
- Programovací jazyk Java v 6. verzi představil balíček javax.script
- Nástroj Qt C++ obsahuje modul QtScript, který interpretuje **JavaScript** stejně jako Java balíček javax.script.

Javascritp lze do HTML zapisovat několika způsoby podobně jako CSS.

Lze využít tag <script> a zapsat tak skript přímo do proudu HTML.

```
<script>
 alert('Hlavu vzhůru, bude hůř!');
</script>
```

Nebo připojit externí soubor se skriptem.

```
<script src="externi_skript.js"></script>
```

Další variantou je in-line zápis

```
<p><a href="#" onClick="alert('Hello');">Click Me</a></p>
```

Nejčastěji se však využívá kombinovaného zápisu. Externím skriptem jsou definovány funkce, normálním zápisem (pomocí <script>) jsou inicializovány proměnné a startovní funkce a in-line skripty volají funkce podle událostí v závislosti na reakcích uživatele.

Javascript se na webových stránkách používá především pro zavádění stránky – odlišení prohlížečů, vkládání menu ze souboru, deklarace funkcí, document.write(), případně jako reakce na uživatelskou událost např. přejetí určitého elementu myší, kliknutí, změna velikosti okna, vyplnění formuláře…

5. JavaScript – pokračování

JavaScript patří mezi objektově orientované programovací jazyky. Podportuje tedy klasický objektový model.

- objekt.metoda() – volání metody (funkce), příkaz, který něco dělá
- objekt.vlastnost – odkazuje na vlastnost daného objektu, má hodnotu, ale nic nedělá
- objekt.podobjekt – odkaz na vnořený objekt

JavaScript má přístup

- K objektům okna prohlížeče (třída Window)
- K prvkům stránky
- K objektům Math a Date, string
- K vytvořeným objektům

Třída Window (objekt Window) je vrcholem hierarchie objektů (tříd). Jeho podobjekty jsou:

- location – adresa načteného dokumentu
- history – historie prohlížení stránek
- navigator – informace o verzi a typu prohlížeče
- screen – vlastnosti obrazovky (šířka, výška, barvy)
- frames – práce s rámy (frame, frameset)
- event – události myši, klávesnice
- document – obrázky, formuláře, odkazy, barvy, jednotlivé HTML elementy…

Nejvyužívanější třídou je třída document, která umožňuje manipulaci s HTML (i jiným typem) dokumentem.

Její nejdůležitější metody jsou:

- document.images
- document.forms
- document.applets
- document.links
- document.anchors
- document.all
- document.frames
- document.styleSheets
- document.scripts
- document.selection
- document.getElementById()
- document.getElementsByTagName()

Třída Math slouží pro použití vyšší matematiky.

Obsahuje metody:

- abs(x)
- exp(x)
- log(x)
- max(x,y)
- min(x,y)
- pow(a,x)
- random()
- sqrt(x)
- ceil(x)
- floor(x)
- round(x)
- acos(x)
- asin(x)
- atan(x)
- atan2(x,y)
- cos(x)
- sin(x)
- tan(x)
- Math.E
- Math.LN10
- Math.LN2
- Math.LOG10E
- Math.LOG2E
- Math.PI
- Math.SQRT2
- Math.SQRT1_2 – odmocnina z 1/2

Třída Date slouží pro práci s datem a časem (vytváření různých odpočítávání, kalendářů...)

Hlavní metody třídy Date:

- get/setFullYear()
- get/setMonth()
- get/setDate()
- get/setDay()
- get/setHours()
- get/setMinutes()
- get/setSeconds()
- get/setMilliseconds()
- get/setTime()

Třída String pracuje s textovými řetězci. Na textové řetězce lze aplikovat vlastnost length, která vrací délku řetězce.

Hlavní metody pro práci s řetězci jsou:

- toUpperCase()
- toLowerCase()
- toString()
- charAt(n)
- charCodeAt(n)
- substring(a, b)
- substr(a, b)
- concat(řetězec1, řetězec2, řetězecN)
- fromCharCode(kód1, kód2, ..., kódN)
- indexOf(podřetězec)
- lastIndexOf(podřetězec)
- split(oddělovač)

6. XML a Json

XML neboli eXtensible Markup Language je značkovací jazyk standardizovaný konsorciem W3C. XML lze označit za standardní formát pro výměnu informací s mezinárodní podporou a vysokým informačním obsahem. Lze jej snadno převést do jiných formátů, existuje automatická kontrola struktury dokumentu a podporuje hypertext a odkazy.

Efektivita XML je silně závislá na struktuře, při špatně navržené struktuře je XML dokument nečitelný a neefektivní.

Každý XML dokument

- Musí mít právě jeden kořenový (root) element.
- Neprázdné elementy musí být ohraničeny startovací a ukončovací značkou. Prázdné elementy mohou být označeny tagem „prázdný element“.
- Všechny hodnoty atributů musí být uzavřeny v uvozovkách – jednoduchých (') nebo dvojitých ("). Opačný pár uvozovek může být použit uvnitř hodnot.
- Elementy mohou být vnořeny, ale nemohou se překrývat; to znamená, že každý (ne kořenový) element musí být celý obsažen v jiném elementu.

Příklad XML

```
<? xml version = "1.0" encoding = "UTF-8"?>        XML declaration
< directory >                                              Root element
< person >                                                      Nested element 1
< name > Adam < / name >                                   Nested Element 1.1
< phone > 777 777 777 < / phone >                          Nested element 1.2
< email > adam@adam.com < / email >                        Nested element 1.3
< / person >                    Exiting the nested element 1
< person >                                                 Nested element 2
< name > Bara < / name >                              Nested element 2.1
< phone > 666 999 666 < / phone >                     Nested element 2.2
< email > bara@bara.com < / email >                   Nested Element 2.3
< / person >                                               Exiting nested element 2
< / directory >                                                 Exiting the root element
```

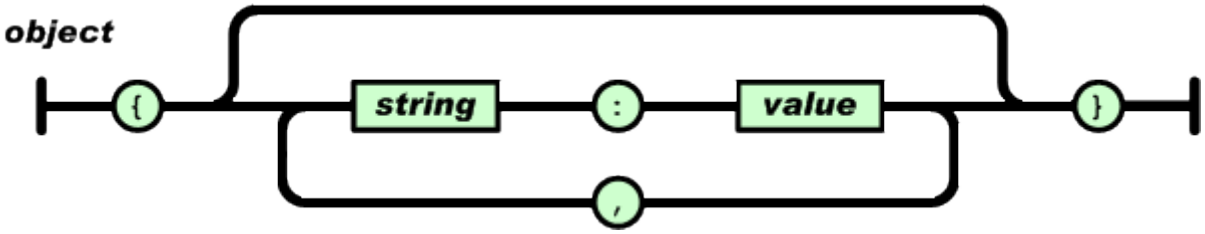
Json – JavaScript Object Notation – je odlehčený formát pro výměnu dat, jednoduše čitelný i zapisovatelný člověkem, zapsaný v textovém formátu a zcela nezávislý na jazyce.

Je založený na dvou strukturách

- Kolekce párů – název/hodnota
    - Realizace: Object, Record, Struct, Dictionary, Hash table, Keyed List, Associative array
- Seřazený seznam hodnot
    - Realizace: array, vector, list, sequence

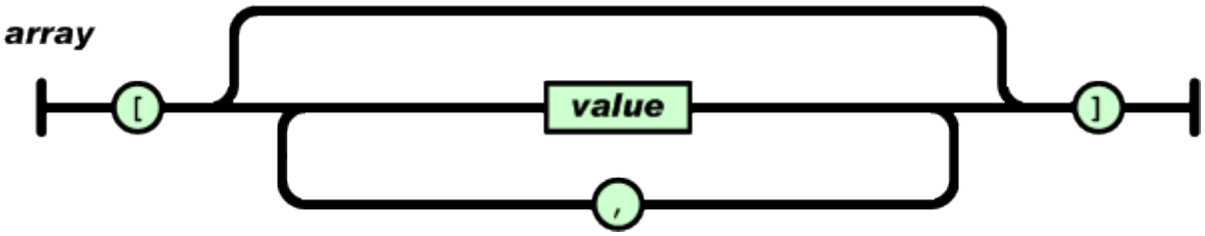
Objekt – neuspořádaná množina párů Název/Hodnota

Zápis: {název1:hodnota1, název2:hodnota2}

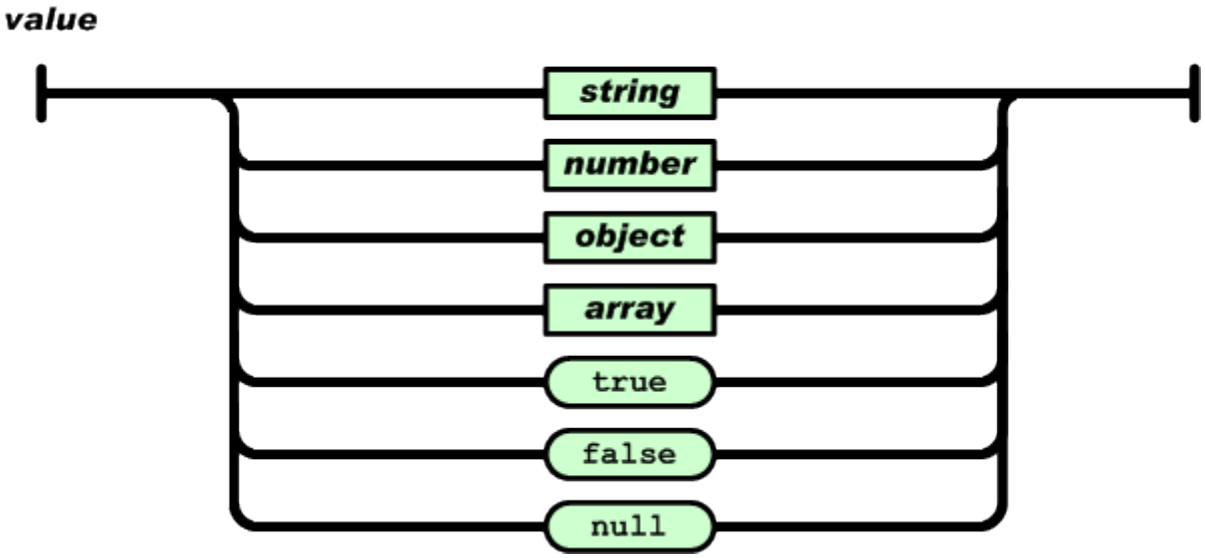


Pole – seřazená kolekce hodnot

Zápis: [hodnota1, hodnota2, hodnota3]

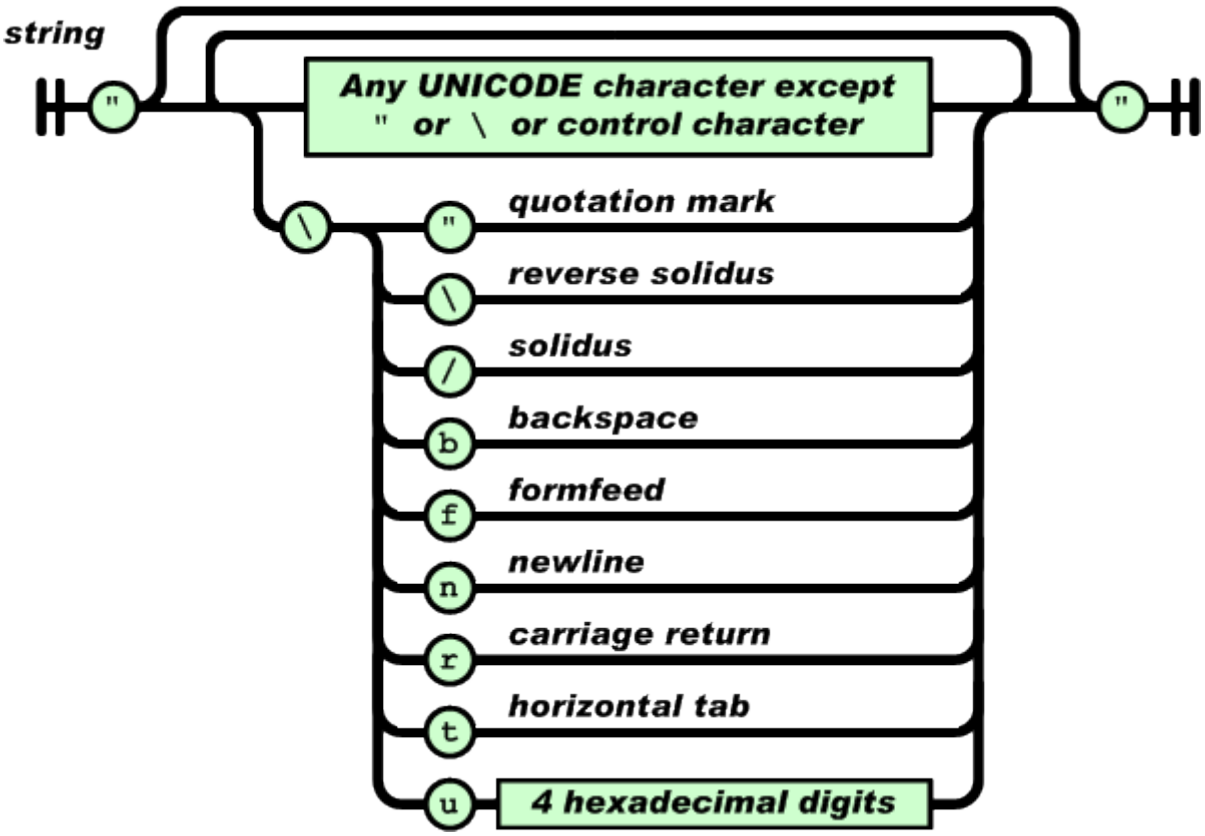


Hodnota – řetězec uzavřený do dvojitých uvozovek, číslo, true, false, null, objekt nebo pole. Tyto struktury mohou být vnořovány

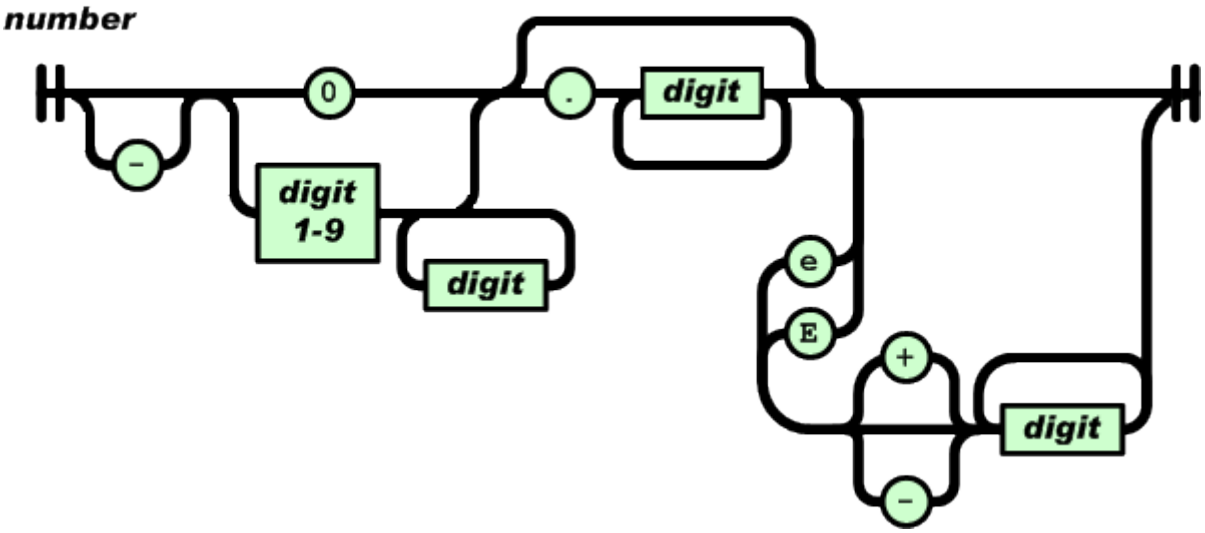


Řetězec – nula nebo více znaků Unicode, uzavřených do dvojitých uvozovek a využívající escape sequence s použitím zpětného lomítka.

Znak – řetězec s jediným znakem



Číslo – podobné číslům z jazyků C a Java, není používán oktalový ani hexadecimální zápis



7. Serverové části webových technologií

Nejprve je nutné říci, co je to webový server.

Za webový server lze označit počítač, který je odpovědný za vyřizování požadavků HTTP(S) od klientů (nejčastěji webových prohlížečů). Vyřízením požadavků se rozumí odeslání cíle specifikovaného URL (typicky webová stránka, ale též statický text, obrázek či jiný soubor). Webové stránky jsou obvykle dokumenty v jazyku HTML. Nebo počítačový program, který provádí činnosti popsané výše (démon).

HTTP je internetový protokol určený pro výměnu hypertextových dokumentů ve formátu HTML, používá tzv. jednotný lokátor prostředků (URL, Uniform Resource Locator), který specifikuje jednoznačné umístění nějakého zdroje v Internetu. HTTP neumožňuje šifrování ani zabezpečení integrity dat, k tomu slouží HTTPS.

HTTPS je protokol umožňující zabezpečenou komunikaci v počítačové síti, využívá protokol HTTP spolu s protokolem SSL nebo TLS.

Výhody HTTPS jsou

- ověření identity
- důvěrnost přenášených dat
- integrita obsahu
- možnost využití HTTP/2 protokolu
- zvýhodnění ve vyhledávači Google

Nevýhody:

- pokles výkonu u staršího hardwaru
- potřeba certifikátu a jeho obnovování
- neumožňuje blokování konkrétních URL pouze blokování celého webu
- mírně složitější konfigurace webového serveru
- možné komplikace u starších webových prohlížečů

Každý webový server je připojen k počítačové síti a přijímá požadavky ve tvaru HTTP. Požadavky vyřizuje a počítači, vrací odpověď, obvykle nějaký HTML dokument. (případně text, obrázek apod). Odpověď serveru je ve tvaru HTTP, je uvozena hlavičkou obsahující stavový kód, za níž následuje samotný obsah.

Odpovědi webového serveru (stavový kód)

- 2xx – úspěšné vyřízení požadavku
- 3xx – problémy spojené s přesměrováním
- 4xx – chyby související s vyřízením požadavku (stránka není dostupná, apod.)
- 5xx – interní chyby serveru

Zdrojem informací serveru může být:

- Statický obsah – předem připravené datové soubory (HTML stránky), výrazně rychlejší než dynamický.
- Dynamický obsah – na základě požadavku jsou shromážděna data (přečtena ze souboru, databáze, nebo nějakého koncového zařízení), zformátována a připravena k prezentaci ve formátu HTML a poskytnuta webovému prohlížeči
- Dynamické vytváření obsahu – řada různých technologií (Perl, PHP, ASP, ASP.NET, JSP, Python apod.), lze poskytovat mnohem větší obsah informací a lze reagovat i na různé „ad hoc“ dotazy klientů

V praxi se kombinují oba přístupy – například cachování, node.js,...

V reálném provozu může docházet k přetížení webového serveru.

Příznaky přetížení:

- pomalá odezva serveru (od jednotek po stovky s)
- Chyby 500, 502, 503, 504
- TCP spojení je nuceno se restartovat ještě před tím, než přijde odpověď
- server odešle nekompletní obsah (toto chování je většinou způsobeno chybou)

Důvody přetížení:

- Klasické přetížení (příliš mnoho lidí se připojí ve stejný čas, ale ne z důvodu útoku)
- DDoS útok, Počítačový vir, který napadne mnoho počítačů a donutí je se připojit
- Internetový bot
- Přetížení fyzické sítě
- Obsah je rozložený na více serverech a některý z nich není dostupný. Všechny dotazy musí obsloužit jen jeden server

Techniky pro zamezení přetížení:

- kontrola síťového provozu pomocí firewallů, HTTP traffic managerů a traffic shapingu
- použití webových cache
- použití rozdílných doménových jmen pro statické a dynamické dotazy
- použití rozdílných doménových jmen a/nebo počítačů pro oddělení velkých souborů, aby ty malé mohly být uloženy v cache
- použití více webových serverů na jednom počítači, každý s vlastní síťovou kartou
- použití více počítačů propojených dohromady a navenek se jevících jako jeden velký server
- přidání více hardware (RAM,CPU)
- vyladění použitého software

8. Serverové části webových technologií II

Webových serverů existuje několik typů. Nejčastějšími jsou:

- Apache HTTP server
- IIS – Internet Information Service
- nginx
- GWS – Google Web Server

Apache HTTP server je softwarový webový server s otevřeným kódem pro GNU/Linux, BSD, Solaris, Mac OS X, Microsoft Windows a další platformy.

Podporuje velké množství funkcí – kompilované moduly rozšiřující jádro, podpora programovacích jazyků na straně serveru (Perl, Python, Tcl, PHP...), různá autentizační schémata(mod_access, mod_auth, mod_digest, a mod_auth_digest), podpora SSL, TLS (mod_ssl), proxy modul (mod_proxy), URL rewriter známý jako rewrite engine z modulu mod_rewrite, configurace souborů logu (mod_log_config) a filtrace (mod_include a mod_ext_filter)

Obsahuje externí modul pro kompresi dat webových stránek (mod_gzip), open source modul pro ochranu a prevenci webových aplikací před napadením (mod_security)

Logy mohou být analyzovány pomocí browseru a skriptů jako AWStats/W3Perl nebo Visitors.

Podporuje mnoho grafických prostředí (GUI) a virtuální hosting – jedna instalace Apache na jednom fyzickém počítači obsluhuje více webových stránek

IIS – Internet Information Service je softwarový webový server s kolekcí rozšiřujících modulů, vytvořený společností Microsoft pro operační systém Windows.

Podporuje řadu protokolů – HTTP, HTTPS, FTP, FTPS, SMTP a NNTP

Moduly pro IIS 7.5

- FTP Publishing Service – Publikování obsahu bezpečně na IIS 7 servery s SSL ověřování a přenos dat.
- Administration Pack – Podporu pro správu UI funkcí správy ve službě IIS 7, včetně oprávnění ASP.NET, vlastní chyby, FastCGI konfiguraci a filtrování požadavků.
- Application Request Routing – Poskytuje proxy-směrovací modul, který předá HTTP požadavky na obsahové servery, založené na HTTP hlavičce serverových proměnných a vyrovnávacích algoritmech.
- Database Manager – Umožňuje snadnou správu místní a vzdálené databáze v rámci služby IIS Manager.
- Media Services – Spojuje mediální platformu s IIS k řízení a spravování multimédií a dalšího webového obsahu.
- URL Rewrite Module – Poskytuje přepisovací mechanismus, jenž změní URL žádost před tím, než jsou zpracovány na webový server.
- WebDAV – Umožňuje autorům webu publikovat obsah bezpečně na IIS 7 servery.
- Web Deployment Tool – Synchronizuje IIS 6.0 a IIS 7 servery. Mění IIS 6.0 na IIS 7

Nginx je softwarový webový server s load managment a reverzní proxy s otevřeným zdrojovým kódem. Pracuje s protokoly HTTP (i HTTPS), SMTP, POP3, IMAP a SSL. Zaměřuje se především na vysoký výkon a nízké nároky na paměť. Je dostupný na Unixu, Linuxu a Unix-like systémech pod BSD, existují varianty pro Solaris, macOS i MS Windows.

Základním cílem Nginx je rychlá distribuce statického obsahu, možnost rozložení zátěže na další servery dle nastavené priority.

Systém umožňuje definovat záložní server, na který Nginx požadavek předá, pokud primární server neodpoví do stanoveného limitu

Příchozí požadavky Nginx asynchronně zpracovává a vyřizuje

Příchozí HTTP (nebo HTTPS) požadavek se nejprve pokusí vyhledat ve své cache (má konfigurovatelnou velikost a dobu uchovávání), pokud jej najde, rovnou odpoví. V opačném případě se obrátí na jeden z definované sady serverů (každý server má definovanou prioritu). Pokud mu server do definovaného času stihne odpovědět, předá odpověď; v opačném případě se obrátí na záložní server (samozřejmě je-li definován). Odpověď, pokud může, uloží do své cache a následující dotazy do vypršení časového limitu životnosti cache vyřizuje právě z cache.

Možnost nastavit limit počtu připojení z jedné IP adresy

Nginx je modulární systém

Jeden z modulů – GEO lokace – umožňuje například dle země předávat požadavky na definované servery, nebo zakázat přístup na stránky z některých zemí

Moduly pro přesměrování dle definovaných pravidel, zabezpečení stránky heslem, podpora komprese gzip, streaming (FLV, MP4)...

9. PHP: Hypertext Preprocessor

Programovací paradigma PHP - imperativní, objektově orientované, procedurální, reflektivní

Vznikl v roce 1995, jeho autorem je Rasmus Lerdorf. První vydání proběhlo 8. června 1995 a zatím poslední verze je 7.2.0 (30. listopad 2017)

PHP se vyznačuje slabou a dynamickou typovou kontrolou.

Hlavní implementace PHP jsou Zend Engine, Phalanger, Quercus, Project Zero, HipHop

PHP je skriptovací programovací jazyk určený především pro programování dynamických internetových stránek a webových aplikací. Skripty jsou prováděny na straně serveru – k uživateli je přenášen až výsledek jejich činnosti. Interpret PHP skriptu je možné volat pomocí příkazového řádku, dotazovacích metod HTTP nebo pomocí webových služeb. Syntaxe jazyka je inspirována několika programovacími jazyky (Perl, C, Pascal a Java)

Jazyk PHP je nezávislý na platformě, rozdíly v různých operačních systémech se omezují na několik systémově závislých funkcí a skripty lze většinou mezi operačními systémy přenášet bez jakýchkoli úprav. Podporuje mnoho knihoven pro různé účely – např. zpracování textu, grafiky, práci se soubory, přístup k většině databázových systémů (mj. MySQL, ODBC, Oracle, PostgreSQL, MSSQL), řadu internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, ...).

PHP je nejrozšířenějším skriptovacím jazykem pro web

Vlastnosti jazyka PHP:

- Jazyk PHP je dynamicky typovaný – datový typ proměnné je vázán na hodnotu, nikoliv na proměnnou.
- Pole jsou asociativní
- Řetězce lze v PHP zapsat 2 různými způsoby:
    - uzavírat do uvozovek (při vyhodnocení se provede nahrazení proměnných uvnitř)
    - uzavírat do apostrofů (nahrazuje se jen escape sekvence \').
- Proměnné lze vytvářet i rušit
- Konstanty lze definovat, nelze je zrušit
- Proměnné mají své úrovně viditelnosti a pravidla pro jejich perzistenci.
- Podporuje reference, pomocí kterých lze do proměnných ukládat odkazy na libovolnou jinou proměnnou, nebo i prvek jejího pole
- Jako reference lze volat i parametry funkce - u každé proměnné eviduje, kolik na ni směřuje referencí, a podle toho se rozhoduje, kdy může kterou proměnnou zrušit.

Výhody:

- PHP je specializované na webové stránky.
- Rozsáhlý soubor funkcí v základní knihovně PHP (přes pět a půl tisíce), další funkce v PECL.
- Nativní podpora mnoha databázových systémů.
- Multiplatformnost (zejména Linux a Microsoft Windows)
- Možnost využití nativních funkcí operačního systému (možná nekompatibilita s jiným OS)
- Strmá křivka učení.
- Obrovská podpora na hostingových službách
- Obrovské množství projektů a kódů, které lze zdarma využít (WordPress, phpBB a další).
- Poměrně slušná dokumentace
- Velmi svobodná licence

Nevýhody:

- Jazyk PHP byl dlouho definován pouze svou implementací, oficiální specifikace jazyka byla oznámena na konci července 2014
- Nekonzistentní pojmenování funkcí
- strpos(), strchr(), ale str_replace(), str_pad().
- Nejednotné názvosloví skupin funkcí
- mysql_XXXX, imap_XXXX, json_XXXX (s podtržítkem) versus imageXXXX, bcXXXX, gzXXXX (bez podtržítka).
- Nejednotné pořadí parametrů, např.: array_map() vs. array_filter().
- Ač jazyk podporuje výjimky, jeho knihovna je používá jen zřídka.
- Slabší podpora Unicode, pouze přes PHP knihovnu (ve verzích po PHP 5 má být Unicode řetězec jako základní typ).
- Ve standardní distribuci chybí ladící (debugovací) nástroj.
- Po zpracování požadavku neudržuje kontext aplikace, vytváří jej vždy znovu (oslabuje výkon).

Jazyk PHP není určen jen pro malé projekty a stránky, lze v něm napsat (naprogramovat) libovolnou aplikaci.

Vybrané významné projekty v PHP:

- MediaWiki – software pro tvorbu webových projektů typu wiki,
- phpBB – balík pro provoz webového fóra
- WordPress – publikační systém pro provoz blogů a podobných aplikací
- Adminer – webová aplikace pro správu databázového systému MySQL
- phpMyAdmin –webová aplikace pro správu databázového systému MySQL
- Texy! – překladač intuitivní syntaxe pro formátování textu na HTML
- Nette Framework – framework pro tvorbu webových aplikací v PHP
- Facebook

10. PHP II – Syntaxe

PHP skript lze v HTML označit několika způsoby

- <? [PHP kód] ?>
- <?php [PHP kód] ?>
- <SCRIPT LANGUAGE="php"> php [PHP kód] </SCRIPT>

Jednotlivé instrukce (příkazy) se oddělují středníkem. Komentáře se uvozují dvojitým lomítkem nebo mřížkou. Víceřádkové komentáře jsou zapsány mezi lomítko hvězdička – hvězdička lomítko (//, #, /* víceřádkový text */)

Typy proměnných v PHP

- Proměnné
- Logický typ – Boolean – hodnota pravda /nepravda, zapisuje se jako TRUE a FALSE (na velikosti nezáleží)
- Celočíselný typ – Integer – celá kladná i záporná čísla (a nulu) od nějakých -2 bilionů po + 2 bilióny
- Desetinné číslo – Float, Real - s přesností obyčejně na 14 desetinných míst
- Řetězec – String – textové řetězce

Typ proměnné se určuje v okamžiku přiřazení hodnoty, během programu může proměnná svůj typ změnit, ať už díky instrukci v kódu nebo v důsledku nějakého výpočtu. Každá proměnná musí mít jednoznačný název, začíná znakem dolaru ($) a bez mezery následuje pojmenováním. První znak toho pojmenování musí být buď písmeno a-z nebo podtržítko. Nesmí to být číslo ani nic jiného. Názvy proměnných rozlišují mezi malými a velkými písmeny. Pro přířazení se používá znak rovná se (=)

PHP umožňuje definovat tři typy polí:

- Indexovaná
- Asociativní
- Vícerozměrná

Ta se využívají jako seznamy, simulace slovníků a kolekce prvků. S poli můžeme pracovat jako se zásobníky nebo frontami, mohou také představovat stromové struktury (prvkem pole totiž může být pole).

Pole mohou být vracena z funkcí PHP (databáze).

PHP patří mezi OOP jazyky.

```
Class ClassName
{
    var $ VariableName
    function Function Name (parameters)
    {
```

```
                body function
    }
}
```

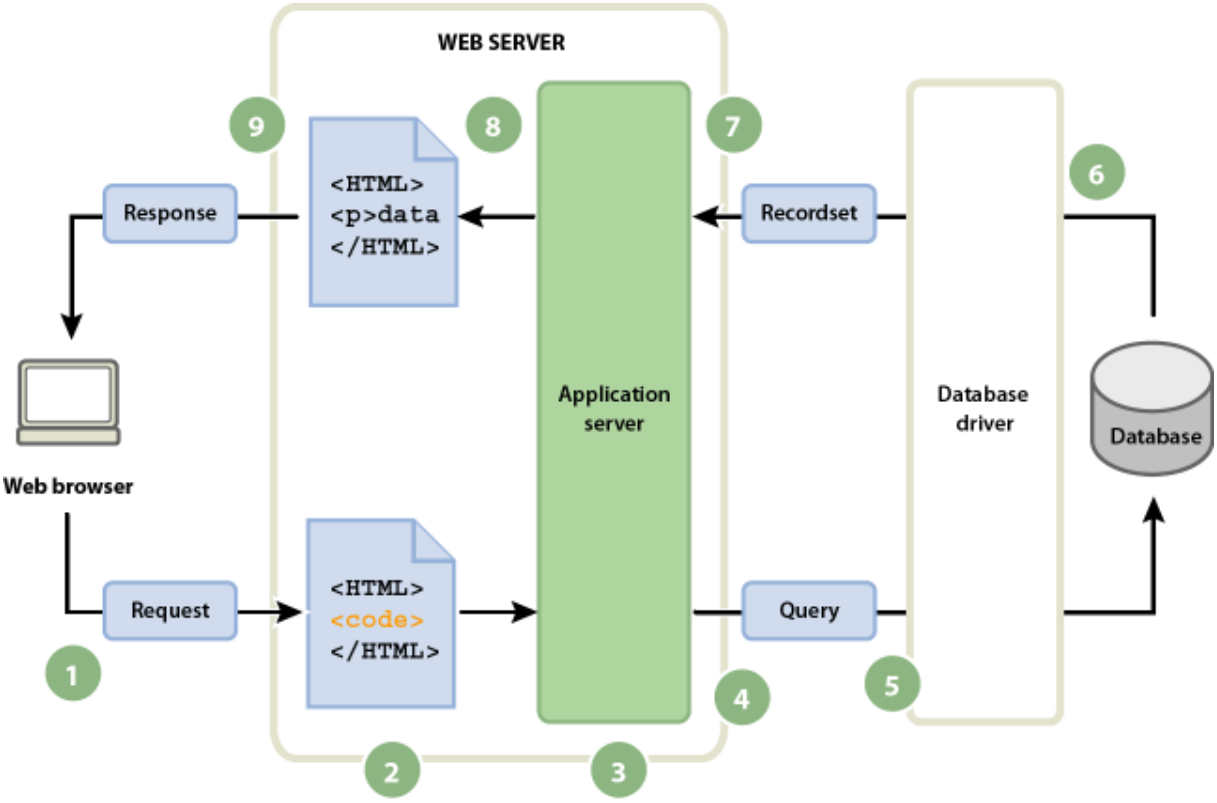
Pozor na dědičnost – PHP nemá privátní metody (funkce).

Navíc PHP umožní zadávat a měnit NEDEKLAROVANÉ atributy!

Mnoho hotových knihoven a částí kódů PHP objekty používá, můžete klidně mít jen jednu "úroveň" objektů bez dědičností. Objekt může existovat bez metod, jen s atributy. Ale někdy je pak lepší použít pole. Někdo zase v PHP pro "přiřazení" každého atributu vytváří metodu nebo metody. Při použití objektů platí více než kdy dříve, že byste si měli vymyslet a dodržovat konvence (třeba, někdo začíná metody, které by se neměly volat zvnějšku objektu podtržítkem).

11. Webové aplikace spolupracující s databázemi

Postup zpracování webového požadavku do databáze.



1. Požadavek – request
2. HTML kód
3. Application server
4. Dotaz – Query
5. Database driver
6. Výstup z databáze
7. Recordset pro server
8. Překlad do HTML
9. Odpověď - Response

Základem pro práci s databází je jazyk SQL.

SQL – Structured Query Language – strukturovaný jazyk pro práci s relačními databázemi

Databáze je systém souborů s pevnou strukturou záznamů, kde soubory jsou mezi sebou navzájem propojeny pomocí klíčů.

Typy databází:

- Hierarchická databáze
- Síťová databáze
- Relační databáze
- Objektová databáze
- Objektově relační databáze

Databázové objekty

- TABLE - základní databázový objektů, který slouží k přímému uložení dat do paměťového prostoru relační databáze
- VIEW - databázový objekt, který uživateli poskytuje náhled na data obsažená v tabulce
- INDEX (KEY) – slouží ke zrychlení vyhledávacích a dotazovacích procesů, definování unikátní hodnoty sloupce tabulky, optimalizaci vyhledávání
- CONSTRAINT – umožňuje vytvořit omezení s podmínkami, jež musí být splněny pro hodnoty jejích sloupců při vkládání nebo změnu záznamů
- TRANSACTION – skupina příkazů, které převedou databázi z jednoho konzistentního stavu do druhého
- TRIGGER – definuje činnosti, které se mají provést v případě definované události nad databázovou tabulkou

SQL příkazy pro manipulaci s daty (DML)

- SELECT – vybírá data z databáze, umožňuje výběr podmnožiny a řazení dat.
- INSERT – vkládá do databáze nová data.
- UPDATE – mění data v databázi (editace).
- MERGE – kombinace INSERT a UPDATE – data buď vloží (pokud neexistuje odpovídající klíč), pokud existuje, pak je upraví ve stylu UPDATE.
- DELETE – odstraňuje data (záznamy) z databáze.
- EXPLAIN – speciální příkaz, který zobrazuje postup zpracování SQL příkazu. Pomáhá uživateli optimalizovat příkazy tak, aby byly rychlejší.
- SHOW - méně častý příkaz, umožňující zobrazit databáze, tabulky nebo jejich definice

SQL příkazy pro definici dat (DDL)

- CREATE – vytváření nových objektů.
- ALTER – změny existujících objektů.
- DROP – odstraňování objektů.

SQL příkazy pro řízení dat (DCL)

- GRANT – příkaz pro přidělení oprávnění uživateli k určitým objektům.
- REVOKE – příkaz pro odnětí práv uživateli.
- START TRANSACTION – zahájení transakce.
- COMMIT – potvrzení transakce.
- ROLLBACK – zrušení transakce, návrat do původního stavu.

Klíčová slova SQL pro dotazování

- TOP – vrátí prvních N řádků
- LIMIT – omezení počtu řádků vrácených příkazem SELECT
- JOIN (FULL LEFT RIGHT INNER CROSS) ON – spojování výsledku dotazu SELECT ze dvou vstupních množin (typicky tabulek relační databáze)
- UNION – sjednocení výsledku dotazu ze dvou nebo více vstupních množin dotazu SELECT
- ORDER BY – seřazení záznamů vybíraných pomocí příkazu SELECT
- WHERE – omezuje výběr řádků z tabulek pomocí podmínek
- GROUP BY – agregace záznamů vybíraných pomocí příkazu SELECT
- WITH ROLLUP – za standardním výpisem se objeví řádek s hodnotou NULL na místo sloupce, podle kterého jsou data agregována (je-li uveden)
- HAVING – umožňuje omezit řádky, které jsou zpracovány agregační funkcí

Databáze a databázové servery

- Microsoft Access
- MySQL
- Oracle
- Microsoft SQL Server
- SQLite

12. DOM – Document Object Model

DOM je objektový model dokumentu, API (aplication programming interface – aplikační programové rozhraní), které definuje obecný standard pro přístup k jakémukoliv platnému HTML dokumentu nebo správně strukturovanému XML dokumentu, je zcela nezávislé na programovacím jazyku

Pomocí DOM lze je obsluhovat jednotlivé prvky (objekty) pomocí JavaScriptu.

Definice DOM popisují jednotlivé úrovně

- Level 0
  - Podpora Intermediate DOM, jenž existoval před vytvořením DOM Level 1. Například DHTML Object Model vyvinutý firmou Microsoft, nebo nepojmenovaný Intermediate DOM od Netscape. Level 0 není formální specifikací publikovanou W3C, ale používá se jako srozumitelná zkratka odkazující na věci existující před standardizačním procesem.
- Level 1
  - Navigace v DOMu (HTML a XML) dokumentu (resp. jeho stromové struktuře) a manipulace s obsahem (včetně přidávání elementů). Specifické elementy HTML jsou obsaženy také.
- Level 2
  - Podpora jmenných prostorů, událostí a filtrovaných pohledů.
- Level 3
  - Standardizovaný mechanismus načítání a ukládání a podpora XML schémat. Umožňuje dynamické vkládání obsahu do dokumentu a přidává nové metody a vlastnosti.
- Level 4
  - Sloučení předchozích standardů DOM Level 3 Core, Element Traversal, Selectors API Level 2, DOM Level 3 Events a DOM Level 2 Traversal and Range a jejich zjednodušení a přiblížení již existujícím standardům, především specifikacím JavaScriptu a HTML5. Specifikace dále zjednoduší časté DOM operace.

Základní myšlenkou DOM je považovat HTML elementy za objekty. Pak každý objekt má vlastnosti – atributy – a může reagovat na události.

Každý objekt musí být identifikován, to je nezbytné pro změnu vlastnosti nebo obsahu pomocí skriptu. Identifikace probíhá pomocí id nebo name.

DOM – univerzální vlastnosti a metody pro procházení a čtení (použití: document.Nazev)

- documentElement - vrací kořenový (root) element dokumentu
- getElementsByTagName() - vrací pole všech elementů daného jména
- parentNode - vrací rodičovský objekt daného objektu
- nextSibling - vrací následujícího sourozence aktuálního objektu, pokud takový existuje, jinak vrací null
- previousSibling - vrací předchozího sourozence aktuálního objektu, pokud existuje
- firstChild - Vrací první dítě daného objektu
- lastChild - vrací poslední dítě daného objektu
- childNodes[] - vrací pole všech objektů (nodelist), které jsou dětmi daného objektu
- nodeName - vrací jméno objektu
- nodeValue - vrací hodnotu (obsah) daného objektu
- data - vrací hodnotu (obsah) objektu typu Text
- className - vrací hodnotu atributu class (pouze pro HTML)
- id - vrací hodnotu atributu id objektu, který musí být typu element, pouze pro HTML
- title - vrací hodnotu atributu title daného elementu, pouze HTML
- item() - vrací určitou položku pole objektů (nodelist) specifikovanou indexem

DOM – metody pro manipulaci s uzly

- createElement(jméno) - vytvoření nového elementu
- setAttribute(jméno, hodnota) - nastavení atributu
- createTextNode(hodnota) - Vytvoří textový uzel
- splitText(pozice) - rozdělení textového uzlu na uzly dva
- normalize() - Sloučí sourozenecké uzly typu Text do jednoho uzlu
- appendChild(objekt) - Přidá potomka uzlu
- insertBefore(objekt, objekt) - 2 parametry – první je uzel, který budeme vkládat, a druhý je uzel, před který budeme vkládat
- cloneNode(true|false) - vytvoří kopii daného objektu
- replaceChild(objekt, objekt) - 2 parametry – první parametr je uzel, který nahradí uzel uvedený jako druhý parametr
- removeChild(objekt) - aplikuje na rodiče uzlu, který má být odstraněn
- removeAttribute(jméno) - Odstraní atribut daného uzlu

DOM lze také využít pro efektivní tvorbu dynamických tabulek, případně pro dynamickou úpravu formátování stránky pomocí CSS.

INFORMATIK - DEUTSCH

Interreg
EUROPÄISCHE UNION
Österreich-Tschechische Republik
Europäischer Fonds für regionale Entwicklung

EINLEITUNG

Das Fachbuch „Lernmaterialien für den Bereich Informatik“ wurde im Rahmen des Projekts "Methodisches Konzept zur effektiven Unterstützung von Schlüsselkompetenzen unter Verwendung einer Fremdsprache - CLIL als Lehrstrategie an einer Hochschule" mit finanzieller Unterstützung des EU-Programms INTERREG VA Österreich - Tschechien 2014 - 2020 erstellt.

Das Projekt wurde in Kooperation zweier technisch orientierten Hochschulen realisiert, dem Institute of Technology and Business in České Budějovice, Tschechien, und der Fachhochschule Oberösterreich. Eines der Hauptergebnisse des Projekts war die Erstellung didaktischer Materialien für vier Fachdisziplinen (Informatik, Logistik und Transport, Maschinenbau und Bauwesen), die an den Partnerinstitutionen in drei Sprachen unterrichtet werden sollen: Tschechisch, Deutsch und Englisch. Als Lehrmethode wurde CLIL (Content and Language Integrated Learning) gewählt, da es den Unterricht von fachspezifischen Inhalten mit einer Fremdsprache kombiniert. So sind die vorbereiteten Materialien nicht nur als Lehr- und Lernmaterial für Lehrende und Studierenden an den oben genannten Hochschulen von großer Bedeutung, sondern können auch von Personen bestimmter Fachbereiche und Mitarbeitenden von Unternehmen im Grenzraum genutzt werden, die so die Möglichkeit haben ihre fachlichen Sprachkenntnisse zu verbessern.

Bei der Aufbereitung der Materialien waren sowohl Lehrende aus den Partnerinstitutionen als auch Praktiker*innen und Expert*innen aus beiden Grenzregionen beteiligt. Die Materialien im Bereich Informatik wurden von Lehrenden beider Partnerhochschulen gemeinsam erstellt. Die jeweiligen Themen wurden in Zusammenarbeit mit Expert*innen ausgewählt. Im Zuge dessen wurden insgesamt zwölf Themenbereiche identifiziert und erstellt: Theorie der Informatik, Algorithmen und Datenstrukturen, Datenbanken, Informations- und Kommunikationstechnologien, Einführung in die JAVA-Programmierung, Objektorientierte JAVA-Programmierung, Einführung in die Medienwissenschaft, IT-Security, Netzwerke, Softwareentwicklung und -Modellierung, Grundlagen der Webanwendung soiwe Web-Technologien. Die Themen wurden so gewählt, dass sie den Bedürfnissen der Praxis entsprechen und ein möglichst breites Spektrum abdecken, von den Grundlagen und der Theorie bis hin zu spezifischen Fragestellungen. Darüber hinaus ist jedes der Themen in Unterkapitel unterteilt, so dass es möglich ist, die Module als Ganzes zu lernen oder nur bestimmte Kapitel zum (Selbst-)Studium auszuwählen. Die vorbereiteten Materialien sind online verfügbar, so dass Studierende und Lehrende die Inhalte und Materialien nach spezifischen Bedürfnissen eigenständig zusammenstellen können.

Die Lehr- und Lernmaterialien wurden, wie bereits erwähnt, in drei Sprachen erstellt. Jeder Themenbereich wurde anschließend von Linguist*innen übersetzt und bearbeitet, um den Prinzipien der CLIL-Methode zu entsprechen und nicht nur den Erwerb von fachlichen, sondern auch sprachlichen Fähigkeiten zu ermöglichen. Derzeit scheint die Kenntnis einer Fremdsprache entscheidend für die Suche nach einem geeigneten Arbeitsplatz in der Grenzregion zu sein. Diese Publikation kann somit nicht nur für Lehrende und Studierende, sondern auch für Absolvent*innen und Mitarbeitende von Unternehmen der jeweiligen Disziplinen innerhalb und außerhalb des grenzüberschreitenden Raums dienen, was einen erheblichen Mehrwert darstellt.

THEORIE DER INFORMATIK

1. Grundlegende mathematische Konzepte und numerische Systeme

- Daten oder Angaben, welche wir mit den Sinnen erfassen können.
- Informationen sind Daten die wir verstehen, Sinn ergeben und quantifizierbar (messbar) sind nach Umwandlung in Zahlen.
- Informationen können sein: Text, Bild oder Ton.
- Die Informatik ist eine Wissenschaft der Speicherung, Verarbeitung unter Verwendung von Daten und Informationen. Es ist Teil der Mathematik und dient als Mittel zur Nutzung der Computertechnologie.
- IT-Branchen:
- Computertechnologie - Untersucht Hardware
- Algorithmisierung - Entwurf von Problemlösungsverfahren
- Programmierung - Konvertierung von Algorithmen in eine Programmiersprache
- Software Engineering - Anwendungsentwicklung
- Computergrafik - Lehre davon, wie man 2D- und 3D-Bilder erstellt und verarbeitet.
- Computermodellierung und Simulation - Anwendung von mathematischen Modellen auf reale Situationen
- Formale Logik, Automatisierungstheorie und Formelsprachen - mathematische Maschinenmodelle und Formelsprachen zum Schreiben von Algorithmen und Programmen
- Kybernetik und Robotik - Entwicklung von Maschinen, die zu eigenständigen Tätigkeiten fähig sind.
- Künstliche Intelligenz - Erforschung der Prozesse des menschlichen Denkens, ihrer mathematischen Beschreibung und Anwendung in der Maschinenentwicklung
- Hardware oder Computerhardware - ein Gattungsname für alle physischen Geräte, mit denen ein Computer ausgestattet ist.
- Software - kann in ein System und eine Anwendung unterteilt werden.
- Informationen, die einem computer gespeichert sind, messen wir in Bit - b - und Byte - B Einheiten.
- Das Bit ist die Basis und die kleinste Informationseinheit. Sie hat zwei Werte: 0, 1
- Byte als Informationseinheit hat einen 8-Bit-Wert.

Da die Informatik Teil der Mathematik ist, verwendet sie auch einige ihrer Begriffe, wie z.B.:

- Kartesisches Produkt
- Beziehung
- Morphismus
- Teilbarkeit
- Der größte gemeinsame Teiler, das kleinste gemeinsame Vielfache.
- Primzahlen

Da der Computer Informationen in Bits und Vielfachen speichert, arbeitet er in einem binären numerischen System. Es ist wichtig über numerische Systeme zu sprechen.

Numerische Systeme werden in positionelle und nicht-positionelle Systeme unterteilt.

Positionsnumerische Systeme - Der Wert jeder Ziffer wird durch ihre Position in der Symbolfolge bestimmt.

Ihr Vorteil ist die hohe Elastizität und ein relativ kleiner Ziffernsatz; der Nachteil ist eine sehr einfache Änderung des Wertes einer Zahl durch einfaches Hinzufügen einer Ziffer zur ursprünglichen Zahl.

Das Hauptmerkmal ist die Basis, dann sind die Gewichte der einzelnen Ziffern die Grundlage der Basis.

Beispiele:

- Unär - die Basis 1
- Binär - die Basis 2, zwei Zustände - ja / nein, 1 Bit, die Symbole 0, 1,
- Oktal - die Basis 8, ein Byte (= 8 Bit), die Symbole 0, 1, 2, 3, 4, 5, 6, 7
- Dezimal - die Basis 10, natürlich für Menschen (zehn Finger), die Symbole 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (die Zahl)
- Hexadezimal - die Basis: 16, 2 Bytes (= 16 Bit), Symbole: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Sexagesimal - die Basis 60, Zeitmessung

Nicht-Positions-Numerische Systeme - Ein Verfahren zur Darstellung von Zahlen, bei dem der Wert einer Ziffer nicht durch ihre Position in einer bestimmten Ziffernfolge gegeben ist. Diese Schreibweisen von Zahlen werden heute kaum noch verwendet und gelten als veraltet, haben aber gewisse Vorteile wie einfache Addition und Subtraktion. Die Nachteile sind, dass sie oft kein Symbol für Null- und negative Zahlen sowie eine lange Anzahl von Zahlen enthielten, die den Wert des größten Symbols des Systems deutlich überschreiten.

Beispiel: Römische Ziffern

Transfer zwischen Systemen

Von einem höheren Basissystem aus

Wenn wir von einem höheren Basissystem zu einem niedrigeren Basissystem wechseln, verwenden wir normalerweise einen Partitionsalgorithmus.

73/2 = 36 + 1 - 1 am 1. Platz von hinten
36/2 = 18 + 0 - 0 auf Platz 2 von hinten
18/2 = 9 + 0 - 0 auf Platz 3 von hinten
9/2 = 4 + 1 - 1 an der vierten Position von hinten
4/2 = 2 + 0 - 0 auf dem 5. Platz von hinten
2/2 = 1 + 0 - 0 bis zur 6. Position von hinten
1/2 = 0 + 1 - 1 an der 7. Position von hinten

Aus einem unteren Basissystem

Der Übergang von einem niedrigeren Basissystem zu einem höheren Basissystem ist einfacher. Wir wissen, dass die Zahl an der n-ten Stelle (berechnet aus 0) die Zahl darstellt, die sich aus der Verstärkung der Basis auf der n-ten Stelle ergibt. Diese Nummern werden nummeriert und übertragen.

$1 \cdot 20 + 0 \cdot 21 + 0 \cdot 22 + 1 \cdot 23 + 0 \cdot 24 + 0 \cdot 25 + 1 \cdot 26 = 73_{10}$

2. Logik

Logik ist die Wissenschaft der richtigen Argumentation oder die Kunst der richtigen Argumentation.

Was ist die Schlussfolgerung (Argument)?

Basierend auf der Wahrheit der Prämisse (Annahmen) P1.... Pn kann geschlossen werden, dass die Schlussfolgerung Z wahr ist.

Wir erkennen verschiedene Arten von Argumentationslogik.

Deduktion

Die Schlussfolgerung Z ergibt sich logischerweise aus den Annahmen 1, ..., Pn, wir bezeichnen P1, ..., Pn | = Z, wenn unter keinen Umständen der Fall eintreten kann, dass die Annahmen wahr wären und die Schlussfolgerung unwahr wäre.

Beispiel:

P1: Alle Kaninchen aus dem Hut sind weiß.
P2: Diese Kaninchen sind aus dem Hut.
Z: ⇒ Diese Kaninchen sind weiß.

Induktion

Generalisierung - von spezifisch bis allgemein

Beispiel:

P1: Diese Kaninchen sind aus dem Hut.
P2: Diese Kaninchen sind weiß.
Z: ⇒ (wahrscheinlich) Alle Kaninchen im Hut sind weiß.

Die Schlussfolgerung Z gilt nur mit einer bestimmten Wahrscheinlichkeit.

Abduktion

Wir erstellen Hypothesen für beobachtete Phänomene, Diagnose von "Störungen".

Beispiel:

P1: Alle Kaninchen aus dem Hut sind weiß.
P2: Diese Kaninchen sind weiß.

Z: ⇒ (wahrscheinlich) Diese Kaninchen sind aus dem Hut.

Die Schlussfolgerung Z gilt nur mit einer bestimmten Wahrscheinlichkeit.

Beispiel:

P1: Alle grünen Fliegenpilze sind stark giftig.
P2: Dieser Pilz ist eine grüner Fliegenpilz.
Z: Dieser Pilz ist giftig.

Gültige Deduktion

P1: Alle grünen Fliegenpilze sind stark giftig.
P2: Dieser Bleistift ist Fliegenpilzgrün
Z: Dieser Bleistift ist giftig.

Richtiges Urteil, Schlussfolgerung Falsch ⇒ Mindestens eine Prämisse ist unwahr.
Das Urteil hat die richtige logische Form.
Logische Konnektive - "und", "oder", "wenn", "falls", "dann" und andere haben feste Bedeutungen, interpretieren sie mit elementaren Aussagen oder Teilen davon (Prädikate und funktionale Begriffe).

Logik ist ein Werkzeug, das hilft, die Beziehung des logischen Ergebnisses zu entdecken, Aufgaben wie "Was bedeutet das?" zu lösen. Es hilft unserer Intuition, die manchmal scheitern kann, weil Prämissen komplex formuliert, verflochten und negiert werden können, die Beziehung des Auftretens ist auf den ersten Blick nicht offensichtlich.

Beispiel:

P1: Alle Männer mögen Fußball und Bier.
P2: Einige Bierliebhaber mögen Fußball nicht.
P3: Xaver mag nur Liebhaber von Fußball und Bier.
Z: Einige Frauen mag Xaver nicht.

Im Wesentlichen, wenn alle Annahmen wahr sind, dann muss die Schlussfolgerung wahr sein.

Ist das Urteil gültig?

Natürlich, wenn Xaver nur Fußball- und Bierliebhaber mag (3.), mag er einige Bierliebhaber nicht (diejenigen, die Fußball nicht mögen (2.)), mag er (nach 1.) einige "Nicht-Männer", d.h. Frauen nicht.

Sie ist jedoch unter der Definition nicht gültig.

Die Beurteilung ist im Bedarfsfall gültig, d.h. unter allen Umständen (Interpretationen), wenn wahre Annahmen wahr und wahr sind.

Aber: In unserem Beispiel müssten diejenigen, die keine Männer sind, nicht als Frauen interpretiert werden. Es gibt hier keine Prämisse, "wer kein Mann ist, ist eine Frau", ebenso brauchen wir noch die Prämisse "wer ein Liebhaber von etwas ist, das er mag".

Wir müssen alle notwendigen Annahmen treffen, um die Schlussfolgerung zu ziehen.

P1: Alle Männer mögen Fußball und Bier.
P2: Einige Bierliebhaber mögen Fußball nicht.
P3: Xaver mag nur Liebhaber von Fußball und Bier.
P4: Wer kein Mann ist, ist eine Frau.
P5: Wer etwas liebt, dem gefällt es.
Z: Einige Frauen mag Xaver nicht.

Nun ist das Urteil richtig, es hat eine gültige logische Form.
Die Schlussfolgerung ergibt sich logischerweise aus den Annahmen (sie sind "informativ, deduktiv einbezogen").

Logische Eigenschaften

Ein gültiges (richtiges) Urteil kann zu einer falschen Schlussfolgerung führen.

Verwendung: Nachweis AD ABSURDUM

Monotonie

Wenn ein Urteil gültig ist, führt die Ausweitung des Satzes von Annahmen auf eine weitere Annahme nicht zu einer Änderung der Gültigkeit des Urteils.

Aus den umstrittenen Annahmen wird jede Schlussfolgerung gezogen.

Reflexivität, Übergang

Gültige Beurteilungsschemata

$A \supset B, A \mid= B$

modus ponens
P1: Keine Primzahl teilbar 3
P2: 9 ist teilbar 3
Z: 9 ist keine Primzahl.

A⊃¬B,B|=¬A

Modus Ponens + Transposition
P1: Alle Menschen sind vernünftig.
P2: Der Stein ist nicht vernünftig.
Z: Der Stein ist kein Mensch.

A⊃B,¬B|=¬A

Modus Ponens + Transposition
P1: 12 eine Primzahl wenn nicht teilbar durch 3
P2: 12 ist teilbar 3
Z: 12 ist keine Primzahl.

¬A∨¬B,B,B|=¬A

Beseitigung der Disjunktion
P1: 12 ist keine Primzahl oder nicht teilbar 3
P2: 12 ist teilbar 3
Z: 12 ist keine Primzahl.

3. Mengen und Beziehungen

3.1. Mengen

Definition - eine Zusammenfassung von Objekten, die genau definiert und identifizierbar sind und Teil der Welt unserer Ideen und Gedanken sind; diese Objekte werden Elemente der Menge genannt.

Wir verwenden das folgende Symbol für Mengen:

- Mengen: A, B, C....
- Elemente: a, b, c .....
- Element Sets: $a \in A$
- Für alle x gilt P : $\forall x : P$
- Es gibt x, für die P gilt: $\exists x : P$
- Es gibt nur ein x, das P hält: $\exists! x : P$
- Konjunktionen, Disjunktionen, Negationen: $\wedge$, $\vee$, $\neg$
- Implikationen, Äquivalenzen: $\Rightarrow$, $\Leftrightarrow$
- Summe, Multiplikation: Σ, Π

Die Menge kann wie folgt hergestellt werden

- Aufzählung der Elemente : A=1,2,3,4,5
- Mit Hilfe der charakteristischen Merkmale: $A=a \in N \mid a<6$

Wir definieren eine leere Menge: $\emptyset$, die keine Elemente enthält.

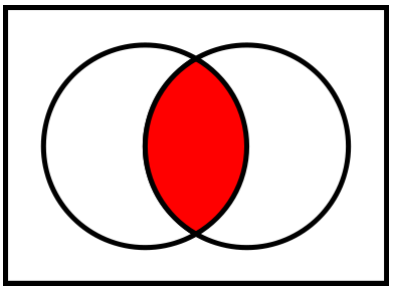
Die Anzahl der Elementelemente bestimmt - Kardinalität - | A | A |, wobei A eine Menge ist.

Wir erkennen Mengen:

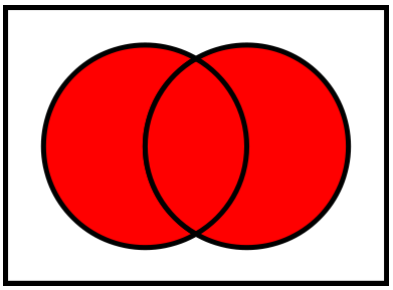
- Endlich
- Unendlich
- Zählbar
- Kontinuum

Grundlegende Operationen mit Mengen

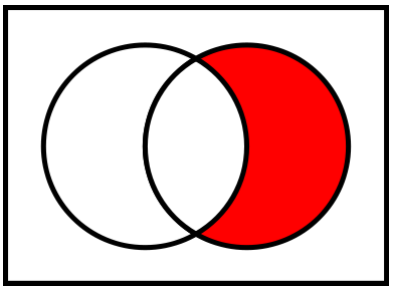
Schnittpunkt: A∩B



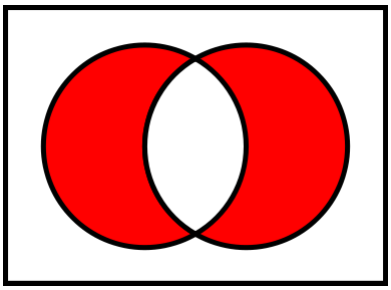
Vereinigung: A∪B



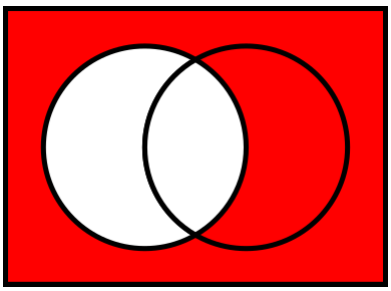
Unterschied der Mengen: B\A



Symmetrische Differenz: A∆B



Komplementär von A in U: A^c=U\A



Teilmenge (Einschließung - das Gegenteil von Ausschluss): A⊆B

(∀x)(x∈A⇒x∈B∈B)

Potentialsatz - Satz aller Teilmengen:

P(∅)=∅
P({a})={∅,{a}}
P({a,b})={∅,{a},{b},{a,b}}

3.2. Beziehungen

Definition: n-te Beziehung zwischen den Mengen A 1, A 2, A 3, ..., A n, wobei n ∈ N, wir meinen jede Untermenge des kartesischen Produkts von n Mengen.

Kartesisches Produkt (diskretes Produkt)

Set X×Y, das alle geordneten Paare enthält, wobei der erste Eintrag von X und der zweite Eintrag z Y ist.

A\B	1	2	3
X	(X,1)	(X,2)	(X,3)
Y	(Y,1)	(Y,2)	(Y,3)
Z	(Z,1)	(Z,2)	(Z,3)

Beziehungen können nach Arität klassifiziert werden zu:

- Unär - jede Teilmenge der Menge N
  Beispiel: Eine positive Zahl ist eine wahr / falsche Aussage.

- Binär - Jeder Satz von geordneten Paaren[x; y]∈M2
  ZB: ist größer als, ist kleiner als, ist eine Teilmenge

- Ternär - Jeder Satz von geordneten Drillingen[x; y; z]∈M3.
  Beispiel: liegt zwischen

- N-te - jeder Satz von geordneten n-Tupletten von Mn.

Operationen mit Beziehungen

Zusätzlich zu den klassischen Mengenoperationen (alle Relationen müssen die gleiche Arithmetik haben), führen wir die umgekehrte Beziehung als: R-1: ∀ a ∈ M1 ∀ b ∈ M2:[a,b] ∈ R ⇔ [b,a] ∈ R-1 und eine zusammengesetzte Beziehung.

Die binäre Beziehung auf dem Set ist:

- Reflexiv, wenn für alle a ∈ M gilt, dass [a, a] ∈ R.

- Symmetrisch, wenn für alle a, b ∈ M gilt, wenn [a, b] ∈ R dann [b, a] ∈ R.

- Antisymmetrisch wenn für alle a, b ∈ M gilt, dass, wenn [a, b] ∈ R und auch [b, a] ∈ R ist, dann a = b.

- Transitiv, wenn für alle a, b, c ∈ M gilt, dass wenn [a, b] ∈ R während [b, c] ∈ R dann [a, c] in R ist.

Zwei besondere Beziehungen

- Äquivalenz ist eine Beziehung, die reflexiv, symmetrisch und transitiv ist.
- Ordnung ist eine Beziehung, die reflexiv, antisymmetrisch und transitiv ist.

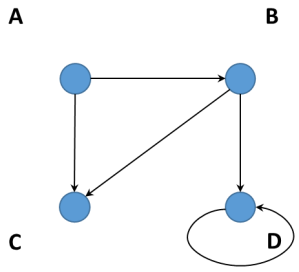
4. Beziehungsstrukturen

Unter relationaler Struktur versteht man eine mathematische Struktur, die zusätzlich zum Trägersatz eine oder mehrere Beziehungen enthält (die beliebige und gleichzeitig unterschiedliche arithmetische Eigenschaften aufweisen können). Dazu gehören orientierte Diagramme und geordnete Mengen (einschließlich ihrer Sonderfälle, linear oder gut angeordnete Linien).

Gerichtete Diagramme

Ein gerichteter Graph G ist ein Paar (V, E), wobei E eine Teilmenge des kartesischen Produkts $V \times V$, $E \subseteq V \times V$ ist.

Die Elemente E werden als Pfeile oder orientierte Kanten bezeichnet. Die orientierte Kante e hat die Form (x, y). Wir sagen, dass diese orientierte Kante auf x basiert und in y endet.
Gerichtete Diagramme werden durch eine Adjazenzmatrix oder eine Liste von Knoten und Kanten dargestellt.



Dazugehörige Matrix:

		WHERE			
		A	B	C	D
FROM	A	0	1	1	0
	B	0	0	1	1
	C	0	0	0	0
	D	0	0	0	1

Für gerichtete Diagramme kann es sein, dass sie nicht symmetrisch sind, 1 ist eine Kan-

te, 0 Kante fehlt.

Der Satz von Eckpunkten und Kanten G: (A, [A, B]), (A,[A,[A, C]) (B,[B, C] D, D])

Geordnete Mengen

Die lineare Anordnung (= Gesamtordnung) ist so definiert, dass:

- Jedes der beiden Elemente der linear geordneten Menge ist vergleichbar, d.h. es besteht eine Beziehung R auf der Menge X und a, b, c ∈ X so dass:
  - Transitivität: (∀x,y,z∈A)((xRy∧yRz)⇒xRz)
  - Schwache Antisymmetrie: (∀x,y∈A)((xRy∧yRx)⇒x=y)
  - Trichotomie: aRb∨bRa∨a=b

Beispiel: Anordnung auf einer Menge von natürlichen und reellen Zahlen

Eine gute Anordnung ist definiert als:

- Der Satz S ist übersichtlich, wenn jeder nicht leere Teil des bestellten Satzes S das kleinste Element hat.

Beispiel: Natürliche Zahlen

Ein geordneter Satz ist einer, auf dem eine Anordnung definiert ist.

Die Anordnung ist eine binäre Beziehung R, die reflexiv, transitiv und schwach antisymmetrisch ist.

(∀x∈A)(xRx) Reflexivität - jedes Element befindet sich in einer R - Beziehung zu sich selbst

(∀x,y,z∈A)((xRy∧yRz)⇒xRz) Transitivität - wenn das Element des Sets in der Anordnung zwischen zwei anderen Elementen liegt, sind die beiden auch vergleichbar.

(∀x,y∈A)((xRy∧yRx)⇒x=y) schwache Antisymmetrie (keine Zyklen in der Anordnung) = unscharfe Anordnung

Scharfe Anordnung - Reflexivität ersetzt durch Antireflexivität - kein Element steht in Beziehung zu sich selbst (∀x∈A)(¬(xRx))

5. Abbildung

Ein spezielles Beispiel für eine binäre Beziehung, die ein eindeutiges Bild zu jedem Muster gewährleistet.

Wenn es sich um eine binäre Relation einer numerischen Menge handelt, dann nennen wir sie Funktion.

Definition:

Abbildung f von der Menge A zu der Menge B ist eine solche binäre Beziehung, bei der jedes Element x in A maximal einem Element y in B zuordnet ist, so dass x,y∈f

Bezeichnung: y=fx f:x→y

Wichtiger Hinweis: y=fx∈B, x∈A

y - Das Ziel des Elements x in der Abbildung f, Funktionswert f im Punkt x
x - Quelle des Elements y in der Abbildung f

Eine Menge von Elementen x∈A, für die es nur ein solches Element y∈B gibt, das y=fx, die Domäne von f genannt wird.

B-Elementsatz y∈B, für den es mindestens ein solches Element x∈A gibt, dass fx=y als der Bereich von f

Wir unterscheiden mehrere Grundtypen von Darstellungen:

Darstellungen von Mengen

f:A→B, A=B

Menge in Menge - die gesamte Menge A ist aus einem Bereich definiert.
F
:A→B, D(f)=A

Von Menge zu Menge (Surjektion) - die gesamte Menge B ist ein Wertebereich.

f:A→B, H(f)=B, ∀b∈B:∃a∈A:f(a)=b

Mengen in Mengen

f:A→B, D(f)=A∧H(f)=B,
∀a∈A:∃!b∈B:b=f(a) ∧∀b´∈B:∃!a´∈A:f(a´)=b´

Einfache (injektion)

$\forall b \in B: \exists! a \in A: f(a)=b$

Gegenseitige eindeutige (Bijektion) - Injektionsfunktion von Set A auf Set B (ist injektiv und surjektiv zugleich)

$f: A \leftrightarrow B$

Invers - Existiert nur für einfache Darstellung

Wenn es $f: A \rightarrow B$ einfach ist, dann wird es dargestellt.

$f^{-1}: B \rightarrow A$, které $\forall b \in H(f)$: $f^{-1}(b)=a \in D(f): y=f(x)$, heißt invers.
$D(f^{-1})=H(f)$, $f^{-1}(y)=x \Leftrightarrow f(x)=y$

Abbildungen können auch zusammengesetzt werden:

Mengen A, B, C

Darstellung: $f: A \rightarrow B$ und $g: B \rightarrow C$

Zusammengesetzte Darstellung: $h: A \rightarrow C$, $h=g \circ f$, $h(a)=g(f(a))$ $\forall a \in A$
Sie ist in der Regel nicht kommutativ, sondern assoziativ.

Umkehrung eines komponierten Morphismus: $(g \circ f)^{-1}=f^{-1} \circ g^{-1}$

6. Boolesche Algebra

Boolesche Algebra ist definiert als: Sechste (A, ∧, ∨, -, 0, 1,) wobei A keine Leere Menge ist, 0∈A - das kleinste Element, 1∈A - das größte Element, - die einzige Operation (Komplement), ∧ der Schnitt, das logische Produkt, ∨ die logische Summe.

Es basiert auf Axiomen:

Kommutativität

$$x \vee y = y \vee x$$
$$x \wedge y = y \wedge x$$

Distributivität

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Neutralität 0 und 1

$$x \vee 0 = x$$
$$x \wedge 1 = x$$

Komplementarität

$$x \vee \bar{x} = 1$$
$$x \wedge \bar{x} = 0$$

A hat die folgenden Eigenschaften:

- Absorption: $x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$
- Aggression von Nullen: $x \wedge 0 = 0$
- Aggression von einem: $x \vee 1 = 1$

- Idempotenz : x v x = x, x ∧ x = x
- Negation der Absorption: x v (- x ∧ y) = x v y, x ∧ (- x v y) = x ∧ y
- Doppelte Negation: - (- x) = x x
- Die Gesetze von De Morgan: - x ∧ - y = - (x v y), - x v - y = - (x ∧ y)
- 0 und 1 ergänzen sich gegenseitig: -0 = 1, -1 = 0

Grundlegende Ein-Eingangs-Operationen:

- ID (Identität)
- NICHT (negieren)
- Zwei grundlegende Funktionen des Eingangs.
- ODER (logische Summe)
- UND (logisches Produkt)

Wahrheitstabelle für grundlegende Funktionen:

A	B	ID(A)	ID(B)	NOT(A)	NOT(B)	A OR B	A AND B
0	0	0	0	1	1	0	0
0	1	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	1	1	1	0	0	1	1

Abgeleitete Zwei-Eingangs-Operationen:

- NOR (Not OR - Negation der Summe)
- NAND (Nicht UND - negiert das Produkt)
- Implikation ⇒ (bei Erfüllung der Annahme A resultiert B, bei nicht erfüllter Erwartung resultiert irgendetwas und damit 1)
- Äquivalenz von EQ ⇔ (Anpassung der Eingänge)
- XOR (Exklusives ODER - Einzigartige Eingänge)

Wahrheitstabelle für abgeleitete Zwei-Eingangsoperationen:

A	B	NOR	NAND	⇒	⇔	XOR
0	**0**	1	1	1	1	0
0	**1**	0	1	1	0	1
1	**0**	0	1	0	0	1
1	**1**	0	0	1	1	0

Nehmen wir den Ausdruck: A (B + NICHT (C))

Dieser Ausdruck kann beschrieben werden mit Hilfe von:

Wahrheitstabellen:

i	A	B	C	A(B+NOT(C))
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Algebraischer Ausdruck:

UND (NICHT (B) (NICHT (C)) + AB (NICHT (C))) + AB (NICHT (C)) + ABC
Dies ist die Summe aller Ausdrücke, die das Ergebnis 1 haben.

Setzen Sie den Zustandsindex so, dass die Zeilen in der Wahrheitstabelle mit der Nummer 0 versehen sind (siehe Spalte i).

A (B+NOT(C)) = {4, 6, 7}

Um den algebraischen Ausdruck zu minimieren, kann Karnaughs Karte verwendet werden.

Es wird so ausgefüllt, dass die Spalten (Zeilen), über denen die Variable angegeben wird, für sie 1 sind.

Die Tabelle zeigt die Werte nach dem Alphabet.

			B	
		C		
	000	001	011	010
A	100	101	111	110

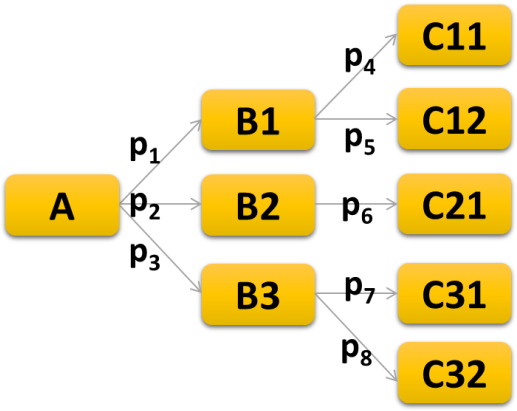
<u>Beispiel:</u> A (NOT (B) (NOT (C)) + A (NOT (B)) C + ABC = A

			B	
		C		
	0	0	0	0
A	1	0	1	1

7. Wahrscheinlichkeit und Statistiken

Der Unterschied zwischen Determinismus und Stochastik besteht darin, dass der Determinismus im Voraus bekannt ist. Von A nach B geht mit 100%iger Wahrscheinlichkeit (ich gehe immer diesen Weg, eine andere Option gibt es nicht).

Im Gegensatz dazu geschieht in den stochastischen Phänomenen alles nur mit einer gewissen Wahrscheinlichkeit, von A bis B gehe ich mit der Wahrscheinlichkeit p 1. Aber ich kann auch zu B2 mit der Wahrscheinlichkeit p 2 und zu B3 mit p3 gehen. Das heißt, ich kenne nur die Wahrscheinlichkeit, mit der das Ergebnis passiert. Zufällige Prozesse spielen in diesem Phänomen eine Rolle. Wenn wir uns das Bild ansehen, muss die Summe von p1, p2 und p3 gleich 1 sein, oder von A gehe ich zu irgendeinem B mit 100% Wahrscheinlichkeit. Ebenso ist die Wahrscheinlichkeit p6 (B2-> C21) gleich eins.



Die Wahrscheinlichkeit ist wie folgt definiert:

Sei F das Experiment E, der Satz von Bedingungen des Experiments Π und der Satz aller möglichen Ergebnisse F. F enthält die Ereignisse 0, A, B..... Ω, wobei 0 ein Ereignis ist, das nie eintritt und Ω ein Ereignis, das immer eintritt. Daher kann das Experiment mit E = (Ω, F) geschrieben werden. Die Menge der unter den gleichen Bedingungen wiederholten Versuchsergebnisse ist N, die einzelnen Ergebnisse sind 1... n und I ist eine Teilmenge der Ergebnisse. Teilmenge A(I) ist Teilmenge von I, so dass das Ergebnis des Experiments ein Ereignis A ist. Die Anzahl der Ergebnisse in der Teilmenge A (I) ist n A (I) und die Menge aller Ergebnisse ist Ω (I) . Dann gelten alle Teilmengen, die ich anwende, wobei P (A) die

Wahrscheinlichkeit des Phänomens A ist:

$$P(A) \approx \frac{n_A(I)}{n_\Omega(I)}$$

Die relative Frequenz des A-Phänomens kann definiert werden als: $\frac{n_A(I)}{n_\Omega(I)}$

Wahrscheinlichkeitseigenschaften - die Wahrscheinlichkeit von A liegt immer zwischen 0 und 1, einschließlich beider Extremwerte.

Die Wahrscheinlichkeit, dass das Phänomen A nicht auftritt, oder die Negation des Phänomens A ist 1-P (A).

Wenn die Phänomene A und B nicht gleichzeitig auftreten, dann ist P A∩B=∅

Eintrittswahrscheinlichkeit des Phänomens A oder B: P(A∪B) ist PA∪B=PA+P(B)

Bedingte Wahrscheinlichkeit - Wahrscheinlichkeit eines Phänomens A, wenn Phänomen B auftritt:

$$(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Bayes-Satz : PA|B·PB=PB|A·PA

Statistiken sind eine Reihe von Konzepten, Regeln und Verfahren, die uns helfen, numerische Informationen im Hintergrund zu organisieren, um die Techniken zu verstehen und fundierte Entscheidungen zu treffen.

Statistisches Wissen wird auf die Daten angewendet, d.h. die Fakten und Informationen aus Beobachtungen (idealerweise aus Experimenten).

Die Daten werden unterteilt in numerische (quantitative) Daten, die das Ergebnis der Messung sind, und kategorische (qualitative) Daten, die in Gruppen unterteilt sind, die auf gemeinsamen Merkmalen basieren.

Die statistischen Basisvariablen sind der Mittelwert (gewichteter Durchschnitt), der Median, der die zentrale Tendenz, den Mittelwert in der gegebenen Menge und den Modus als den häufigsten Wert darstellt. Als nächstes berechnen wir die Varianz und daraus die Standardabweichung als Wurzel. Wir können auch eine Kovarianz finden, die bestimmt, wie sehr sich die beiden Werte zusammen ändern. Und dann Schiefe und Kurtosis, die

die Verteilung der Werte für eine bestimmte Datei aufdecken.

Die Verteilung der Dateiwerte beschreibt die sogenannte Verteilung. Eine der Grundverteilungen ist die Gaußsche (Normal-)Verteilung. Es ist symmetrisch, mit dem gleichen Mittelwert, Modus, Median, Schiefe und Kurtosis sind Null.
Eine der grundlegenden Statistiken ist das Central Limit Theorem, das die Tatsache ausdrückt, dass die Summe vieler unabhängiger Zufallsvariablen dazu neigt, sich in einem kleinen Satz von Verteilungsfunktionen zu verteilen.

8. Informationstheorie

Die Informationstheorie beschäftigt sich mit der Messung, Übertragung, Verschlüsselung, Speicherung und Weiterverarbeitung von Informationen in quantitativer Hinsicht. Ihr Gründer ist C.E. Shannon.

Informationen

Was wir mit der Außenwelt austauschen, wenn wir uns an sie anpassen und sie beeinflussen, durch unsere Anpassung.

Inhalt der Nachricht, Kommunikation, Klärung, Erklärung, Anweisung.

Daten, Zahlen, Zeichen, Befehle, Anweisungen, Befehle, Nachrichten und dergleichen. Betrachten Sie zur Information auch die Ideen und Wahrnehmungen, die von lebenden Organismen empfangen und abgegeben werden.

Die Größeninformationen werden als Ordnung (Unsicherheit) der Elemente (System) bereitgestellt.

Die Systeminformationen sind umso größer, je größer die Wahrscheinlichkeit des Auftretens einzelner Zustände ist. Die Informationen sind größer, wenn die Nachricht etwas Neues enthält, das nicht bekannt war oder nicht leicht erraten werden konnte (es ist unwahrscheinlich).

Für die Weitergabe werden die Informationen benötigt.

Jede Methode der Kodierung = Übertragung auf geeignete Signale oder Symbole

Signal - informationstragende physikalische Größe

Nachricht - Ausdruck von Informationen unter Verwendung einer Folge von Symbolen (Zeichen)

Der Bericht als solcher besteht aus drei Teilen:

- Syntax - Track
- Syntaktische Inhalte
- Es kann selbst ohne semantische und pragmatische Inhalte existieren.
- Sie bezieht sich auf die gegenseitige Anordnung von Zeichen als Informationsträger.
- Beschreibt die Seite mit den quantitativen Informationen.
- Semantik - Informationsgehalt
- Semantischer Inhalt

Die Bedeutung der Nachricht kann nicht gemessen werden.

Eine Nachricht mit der gleichen Informationsmenge kann in verschiedenen Sprachen verfasst werden.

Beschreibt die qualitative Informationsseite

Bedeutung - pragmatische Inhalte

Bestimmt die Bedeutung (Nützlichkeit, Wert) der Nachricht und die Priorität.

Qualitative Informationsseite

Der Informationsgehalt der Nachricht wird durch Entropie gemessen.

Informationen sind ein Maß für die Höhe der Unsicherheit oder Unsicherheit über jede zufällige Handlung, die durch die Realisierung dieses Ereignisses eliminiert wird.

Wir bestimmen den Grad der Unsicherheit als die Entropie des Systems.

$$H(X) = -\sum_{i=1}^{n} p(x_i)p(x_i)$$

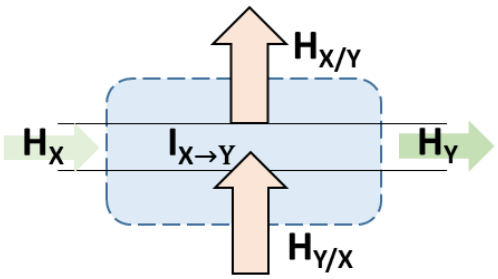
Informationen über System X erhalten Sie bei:

- Direkter Beobachtung
- Indirekt über ein System von Y (X-System ist für uns nicht verfügbar)

Der Systemstatus Y ist nicht unbedingt identisch mit dem Zustand von X (der Text des Telegramms in einer Stadt X, Y in der zweiten Stadt).

Unterschiede zweierlei Art

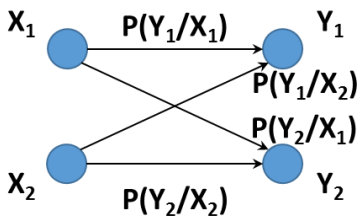
- Einige Zustände des X-Systems erscheinen in einem Zustand Y (Y kann die Feinheiten in X nicht unterscheiden, Y ist gröber als X).
- Übertragungsfehler zwischen X und Y - z.B.: Rauschen
- Verwendung eines Kanals



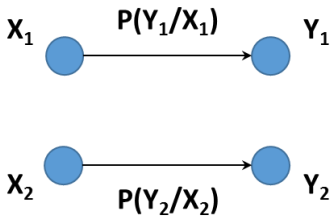
H X / Y - durchschnittlich verlorene Informationen
H Y / X - durchschnittliche Störinformation
I X X → Y - gegenseitige Information

Für Kanalinformationen unterscheiden wir **zwei Arten von Kanälen**:

Laut (störend):



Lautlos:



P (Y 1 / X 1) Wahrscheinlichkeit beim Senden des Elements X 1 empfangen wir Y 1
P (Y 2 / X 2) Wahrscheinlichkeit beim Senden des Elements X 2 empfangen wir Y 2
P (Y 1 / X 2) Wahrscheinlichkeit beim Senden von Element X2 empfangen wir Y 1
P (Y 2 / X 1) Wahrscheinlichkeit beim Senden des Elements X 1 empfangen wir Y 2

Kanalpermeabilität (Kanalkapazität) - Kanalfähigkeit zur Übertragung von Informationen. Maximale Informationen, die pro Zeiteinheit übertragen werden können.

$C = B\left(1 + \frac{S}{N}\right)$, wobei B die Kanalbreite, S das Signal und N das Rauschen ist.

Abtasttheorem

Eine genaue Rekonstruktion eines kontinuierlichen, frequenzbegrenzten Signals aus seinen Abtastwerten ist möglich, wenn die Abtastrate größer als das Doppelte der höchsten harmonischen Komponente des abgetasteten Signals ist.

Die minimal mögliche Länge der Signalelemente

$\tau_0 = \frac{1}{2F_m}$, F ist die Grenzfrequenz, Bandbreite.

9. Anwendung der Informationstheorie

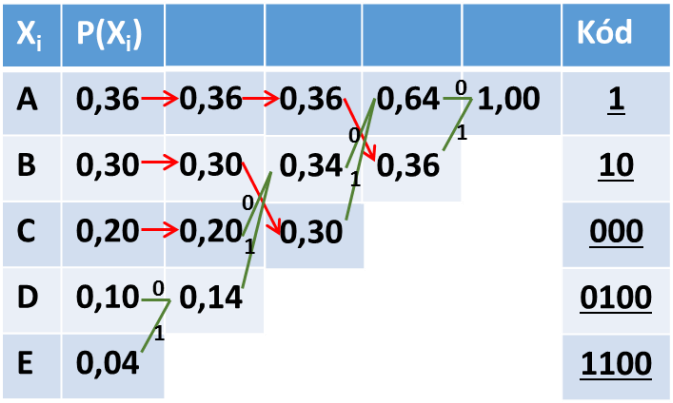
Beispiele sind Huffman-Kodierung, arithmetische Kodierung, LZW, JPEG, MP3, TIFF, Fehlererkennungs- und Fehlerkorrekturcodes, statistische Anwendungen und minimal description length (MDL).

Die Huffman-Kodierung ist ein Algorithmus zur verlustfreien Datenkompression. Konvertiert die Zeichen der Eingabedatei in Bitstrings unterschiedlicher Länge. Die häufigsten Symbole in Bitstrings mit der kürzesten Länge (sogar 1bit). Weniger häufige in längere Ketten (kann länger als 8 Bit sein).

Es besteht aus zwei Phasen

- Übergibt die Datei und erstellt Statistiken.
- Erstellt einen binären Baum, um Daten zu komprimieren.
- Der Code wird von den Blättern bis zur Wurzel erzeugt.

Beispiel:



Die Shannon-Fano-Kodierung ist die statistische Methode der verlustfreien Kompression. Huffman's Kodierung unterscheidet sich nur im Binärbaum-Design. Der Zeichensatz wird rekursiv in zwei etwa gleiche Teilmengen unterteilt. Einem Subcode wird dann eine binäre 1 und der zweite 0 zugewiesen, so dass der Code von der Wurzel bis zu den Blättern aufgebaut ist, im Gegensatz zur Huffman-Codierung, welcher von den Blättern bis zur Wurzel gebildet wird, und deshalb nicht optimal sein muss.

Beispiel:

Char	p(x)	s	Groups				Code
A	0,36	1	0				0
B	0,3	0,64	1	0			10
C	0,2	0,34		1	0		110
D	0,1	0,14			1	0	1110
E	0,04	0,04				1	1111

Die arithmetische Kodierung ist eine Los-Längenkompression von Daten variabler Länge eines Codeworts. Es kodiert den gesamten Text in eine einzige Zahl, einen Bruchteil n∈<0;1). Es ist anstrengend, mit realen Zahlen zu berechnen. Für einen längeren Bericht wären wir nicht mehr in der Lage, die erforderliche Präzision zu erreichen, und einige Subintervalle könnten beginnen zu verschmelzen. Es verwendet eine Blockkompression, bei der die Blöcke groß genug sind, um eine ausreichende Genauigkeit bei der Verteilung zu gewährleisten.

Informationen können während der Übertragung gesichert werden, um Fehler zu erkennen und zu korrigieren und unbefugtes Lesen zu verhindern.

Codes zur Fehlererkennung und –behebung

Parität - Daten Für jedes Segment wird ein anderes Bit verwendet, dessen Wert der Anzahl der binären Komplemente der Zahl ungerade oder gerade ist (ungerade / gerade Parität).
Gerade: 10011011 10010101 → 100110111 1001010101010 100101010
Ungerade: 10011011 10010101 → 100110110 1001010 100101011

CRC - Prüfsumme - Die Daten werden in Abschnitte der erforderlichen Länge (8, 16, 32 Bit) unterteilt und diese Segmente werden ohne Übertragung nach den Bits hinzugefügt. Das resultierende Datensegment verbindet sich mit den übertragenen Daten.

00001101	Data
00010000	
01000100	
10010000	
11110001	**CRC**

Hamming-Code ist ein linearer Code, der in der Telekommunikation verwendet wird, um bis zu zwei fehlerhafte Bits zu erkennen oder ein schlechtes Bit zu reparieren.

Algorithmus

Alle Bitpositionen, die ein Vielfaches von 2 sind, werden für das Paritätsbit (1, 2, 4, 8, 16, 32,....) verwendet.

Alle anderen Bitpositionen gehören zum kodierten Informationswort (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17,....).

Jedes Paritätsbit wird aus einigen der Informationswortbits berechnet. Die Position des Paritätsbits gibt die Reihenfolge der Bits an, die im Codewort gescannt und übersprungen werden.

Für das Paritätsbit p 1 (Position 1) wird im Rest des Codeworts 1 Bit übersprungen, 1 Check, 1 Skip, 1 Check,

Für Paritätsbit p 2 (Position 2) wird das erste Bit übersprungen, 2 Checks, 2 Skips, 2 Checks, etc.

Für p 3 (Position 4) werden die ersten 3 Bits übersprungen, 4 werden überprüft, 4 werden übersprungen, 4 werden überprüft,

Gegen unbefugtes Lesen

- Verschlüsselung – Kryptographie; Historisch
- Stenographie - Verstecken der Botschaft
- Ersetzungschiffre - ersetzt jedes Zeichen durch ein anderes durch eine Regel.
- Zeichenverschiebung - Caesars Chiffre - jeder Buchstabe des Alphabets wurde auf eine feste Anzahl von Positionen verschoben.
- Tabellen der Ersetzungen - Ersetzen eines Zeichens durch ein anderes ohne internen Kontext oder Kenntnis des Schlüssels.
- Vigener Cipher - verwendet Passwörter, deren Zeichen den Offset des offenen Textes bestimmen, so dass der offene Text in Zeichenblöcke unterteilt wird, solange das Passwort gültig ist und jedes Zeichen mit dem entsprechenden Passwort-Zeichen hinzugefügt wird.
- Verman's Chiffre - die einzige bekannte Chiffre, die sich bisher als nicht nachweisbar erwiesen hat, basiert auf der Hinzufügung von offenen Buchstaben und Passwörtern, aber das Passwort ist ein Block von zufällig ausgewählten Daten der gleichen Größe wie der offene Text.

Transpositionsraster

- Skytalé - eine Art von Verschlüsselung, die aus einem Zylinger und einem Wickelpapier oder Pergament darauf besteht, auf das eine Nachricht geschrieben wird.

– Modern

- Symmetrische Verschlüsselung - eine herkömmliche Verschlüsselung, verwendet einen einzigen Schlüssel zum Ver- und Entschlüsseln, DES (Data Encryption Standard), das aktuelle AES (Advanced Encryption Standard).
- Asymmetrische Verschlüsselung - Verwenden Sie verschiedene Schlüssel (privater und öffentlicher Schlüssel), RSA, PGP (OpenPGP) zur Ver- und Entschlüsselung, verwendet für die elektronische Signatur.
- Hash-Funktion - eine einseitige mathematische Funktion, die beispielsweise zur Sicherstellung der Datenintegrität verwendet wird.

Unterzeichnung

- Elektronische Signatur - Kennzeichnung bestimmter Daten, die eine klassische handschriftliche Unterschrift oder eine beglaubigte Unterschrift am Computer ersetzen. Es ist mit einer Datennachricht versehen oder logisch mit ihr verbunden, es ermöglicht die Authentifizierung der signierten Person in Bezug auf die Datennachricht. Eine elektronische Signatur ist ein Mittel zur Überprüfung der Identität des Absenders.

10. Theorie der Komplexität

Die Komplexitätstheorie konzentriert sich auf die Klassifizierung von Rechenproblemen nach ihrer eigenen Komplexität und der Beziehung zwischen den Klassen. Das Problem ist eine Aufgabe, die auf einem Computer gelöst werden kann. Das Problem gilt als schwierig, wenn seine Lösung erhebliche Ressourcen erfordert, unabhängig davon, welcher Algorithmus verwendet wird. Es formalisiert diesen Ansatz und führt Computermodelle zur Untersuchung und Quantifizierung der Menge der Ressourcen durch, die zur Lösung von Problemen benötigt werden (Zeit, Speicher). Eine weitere Komplexitätsstufe ist viel Kommunikation, Gate-Zähler der Schaltung, die Anzahl der Zugriffe auf den Cache und die Anzahl der Prozessoren. Eines der Ziele ist es, die praktischen Grenzen dessen zu bestimmen, was Computer berechnen können und was nicht.

Analyse von Algorithmen vs. Theorie der Komplexität vs. Theorie der Berechenbarkeit

- Die Algorithmusanalyse beschäftigt sich mit der Menge der Ressourcen, die von einem bestimmten Algorithmus benötigt werden.
- Die Komplexitätstheorie identifiziert allgemeinere Fragen zu allen Algorithmen, die zur Lösung eines bestimmten Problems verwendet werden können.
- Der Versuch, Probleme nach den Einschränkungen der verfügbaren Ressourcen zu klassifizieren, führt zu Einschränkungen der verfügbaren Ressourcen.
- Die Quantifizierungstheorie fragt, welche Probleme grundsätzlich algorithmisch gelöst werden können.

Die Grundlage ist ein Rechenproblem

Problem: Unendliche Sammlung von Beispielen und Lösungen für die Situation, Eingabe von Input, d.h. Eine Instanz des Problems kann nicht mit dem Problem selbst verwechselt werden. Das Problem muss unabhängig von seiner Zuordnung angegangen werden.

Beispiel: Prime Number Test - Input: Anzahl; Ausgabe: Ja / Nein

Sie können das Problem auf verschiedene Arten erfassen: Der grundlegendste Weg ist die Form eines ausgesprochenen Satzes. Für die Bearbeitung per Computer ist eine Übersetzung in die Sprache der Computer notwendig. Mathematische Aufgaben in Graphen werden z.B. durch Matrizen festgelegt.

Das Entscheidungsproblem ist die Grundart der Probleme der Komplexitätstheorie. Sie hat nur zwei Ausgaben Ja / Nein. In der Sprache, in der die Mitglieder der Sprache die Fälle mit der Antwort JA und die anderen Mitglieder mit der Antwort NEIN sind, sollten Sie einen Algorithmus verwenden, um zu entscheiden, wie die angegebene Eingabe mit

dieser Sprache übereinstimmt. Wenn JA, dann gibt der Algorithmus die Eingabe zurück und wird akzeptiert, andernfalls abgelehnt. Der Algorithmus berechnet ein charakteristisches Merkmal der Sprache.

Formale Sprachen (und deren Probleme) sind einige der Alphabete, die nicht unbedingt binär sind.

Die Problemfunktion ist ein Rechenproblem, bei dem ein Ausgang der Gesamtfunktion für alle möglichen Eingaben gilt, aber komplexer ist als der Ausgang der Entscheidungsfunktion. Das Problem der Funktion ist umfangreicher als das Entscheidungsproblem. Es kann auch wie folgt auf das Entscheidungsproblem umgestellt werden: Wir wollen zwei Zahlen multiplizieren, die Antwort auf das Entscheidungsproblem ist ein Tripel (a, b, c) und die Antwort JA wird nur ausgegeben wenn gilt: a * b = c.

Für die Komplexitätstheorie müssen Sie die Eingabegröße messen. Die Größe der Eingabe hängt davon ab, wie lange es dauert, den Algorithmus zu verarbeiten. Diese Teilprobleme, die der für die Lösung benötigte Platz sein können, verarbeiten einen separaten Algorithmus und beziehen sich alle auf die Größe der Bit-Eingabe. Die Komplexitätstheorie beschäftigt sich damit, wie der Algorithmus mit der Eingabegröße umgehen wird.

Beispiel: Lösen des verbundenen Graphen von n Kanten im Vergleich zum Graphen von 2 n Kanten. Wenn der Eingang n ist, dann ist die Zeit, die zur Berechnung der Funktion (n) benötigt wird. Wenn die Funktion τ(n) polynomial ist, sprechen wir von einem polynomialen Algorithmus.

Cobham These - das Problem kann in polynomialer Zeit gelöst werden, wenn es für den Algorithmus existiert, der die n-Bit-Eingabe in der Zeit verarbeitet, wobei c eine Konstante ist, die vom Problem abhängt, nicht von seiner Eingabe.

Wir messen aber nicht nur die Eingaben, sondern auch die Ressourcen, sei es ein bestimmter Algorithmus oder ein Problem.

Im Allgemeinen ist die Datengröße n, und nicht für einen bestimmten Eingabewert auf die (Größe n), in der Regel für alle möglichen (unendliche Zahl) → Größe und Komplexität schätzen in der Regel asymptotisch. Die Abhängigkeit von IN-Ressourcen wird gemessen an:

- Verstrichene Zeit (in Schritten)
- Speicher (in Bits / Bytes / Zelle)
- Pakete (in der Regel in Frames)
- Cache (z.B. die Anzahl der Zugriffe)

Complexity Theory Class definiert die Komplexität der Probleme:

- Klasse P
  Das Entscheidungsproblem der U liegt in der Klasse P, wenn es eine Turingmaschine gibt, die die Sprache LU v in Polynomzeit bestimmt.
  Ex ..: Finden des kürzesten Pfades, minimaler Spannweitenbaum

- Klasse NP
  Die Entscheidungsrolle der U liegt in der Klasse NP, wenn und nur wenn es eine nicht-deterministische Turingmaschine gibt, die die Sprache LU v in Polynomzeit entscheidet.
  Ex ..: K-Farbe (Farbpalette kann bis zu den Farben aufgegeben werden ?) Clique des Graphen (existiert Clique im Graphen mindestens von k Knoten?)

- NPC-Klasse - NP-komplettes Problem
  Das Problem ist in der Klasse NP und es ist auch wahr, dass das Polynom die Rolle der einzelnen Klassen NP reduziert. NP - vollständige Aufgaben sind das "schwierigste" aller NP-Probleme.

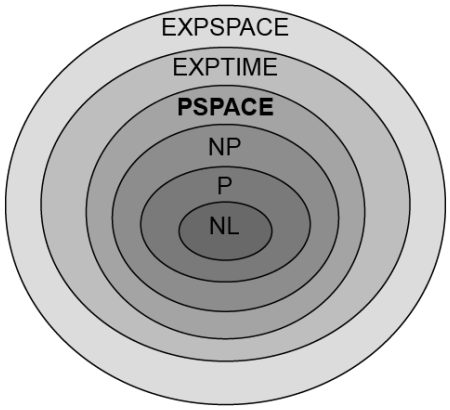
Beispiel: Reisender Verkäufer Problem, Rucksack Problem

Klassen PSPACE und NPSPACE

Die Sprache L ist in der Klasse PSPACE, wenn und nur wenn es eine deterministische Turingmaschine gibt, die mit der Komplexität des Speicherpolynoms arbeitet (d.h. keinen Speicherzellenindex größer als p (n) verwenden) und die Sprache L übernimmt.

Die Sprache L ist in einer Klasse NPSPACE nur dann, wenn es eine nicht-deterministische Turingmaschine gibt, die mit polynomaler Komplexität des Gedächtnisses arbeitet und die Sprache L akzeptiert.

Es hat sich gezeigt, dass NP⊆NPSPACE und des Weiteren entsprechend des Savitch-Satzes gilt PSPACE=NPSPACE.



11. Sprachen und Automaten

Sprachen und Automaten sind der Grundstein der theoretischen Informatik.

Die Sprache bedeutet jede nicht leere Menge V des Alphabets, ihre Elemente sind Zeichen oder Symbole.

Wir definieren:

- Das Wort über dem Alphabet V ist die letzte Folge von Zeichen aus V, w = a1 a2...an
- Wortlänge w - Länge der Sequenz w, | w | w | = n
- Leeres Wort ε - Reihenfolge der Länge 0
- Eine Menge aller Wörter über dem Alphabet V ^ *
- Eine Menge aller nicht leeren Wörter über dem Alphabet V ^ +
- Die Grammatik G ist vierfach (N, Σ, P, S),
- N ist der endgültige Satz von nicht-terminalen Symbolen (Non-Terminals).
- Σ ist der letzte Satz von Terminalsymbolen, so dass kein Symbol gleichzeitig zu N und Σ gehört.
- P ist der letzte Satz von Ableitungsregeln. Jede Regel ist das Formular (ΣUN) ^ * N (ΣUN) ^ * → (ΣUN) ^ *
- S ist ein Element von N, das als Initialsymbol bezeichnet wird.

Konvention

- Wir bezeichnen Terminals - a, b, c, ...
- Kettenklemmen bezeichnet - u, v, w, ...
- Einzelne Nicht-Terminale - A, B, C ..... X, Y, Z
- Zeichenketten von Nicht-Terminals und Terminals - α, β, γ, ...
- Eine leere Zeichenkette wird durch das Symbol e oder sogar ε gekennzeichnet.

Die Grammatik ist nach der **Chomsky-Hierarchie** in mehrere Typen unterteilt.

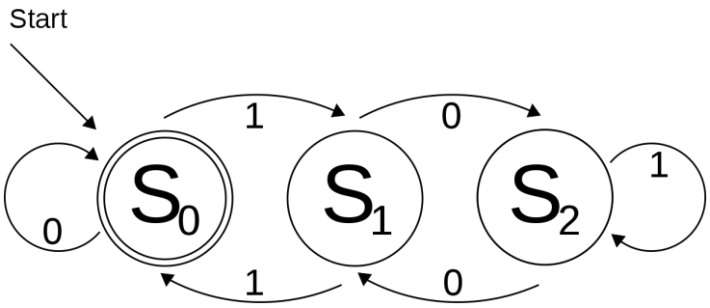
- Typ 0 - Alle formalen Grammatiken (uneingeschränkte Grammatiken)
- Typ 1 - Kontext-Grammatiken
- Typ 2 - Kontextfreie Grammatiken
- Typ 3 - Normale Grammatik

Der endliche Automat ist das theoretische Rechenmodell, das in der Informatik zum Studium formaler Sprachen verwendet wird. Es beschreibt einen sehr einfachen Computer, der sich in einem von mehreren Zuständen befinden kann, zwischen denen er Symbole einschaltet, die aus der Eingabe lesen. Der Satz von Zuständen ist endlich (daher der Name), der endliche Automat hat keinen zusätzlichen Speicher, zusätzlich zum aktuellen Status. Wir unterscheiden deterministische EA - an jedem Punkt in der Tabelle hat

nur einen Zielzustand - und nicht deterministische EA - an jedem Punkt der Tabelle ist nicht ein Zielzustand, sondern der gesamte Satz von Zuständen. Außerdem gibt es in der Übergabetabelle eine leere Eingabespalte, genannt e. Jede nicht-deterministische Maschine kann in deterministisch umgewandelt werden. Der ursprüngliche Satz von Zuständen muss durch seinen potenziellen Satz ersetzt werden. Jeder Zustand der so erzeugten Maschine entspricht einem Satz von Zuständen der ursprünglichen nicht-deterministischen Maschine und es gibt klare Übergänge zwischen ihnen.

Formal ist der endliche Automat definiert als geordnete fünf (S, Σ, σ, s, A) wobei:

- S ist ein endlicher, nicht leerer Satz von Zuständen.
- Σ ist ein endlicher, nicht leerer Satz von Eingabesymbolen, genannt Alphabet.
- σ ist die so genannte Übergangsfunktion (auch eine Übergangs-Tabelle), die die Regeln für den Übergang zwischen Zuständen beschreibt. Kann entweder $S \times \Sigma \rightarrow S$ (deterministischer Automat) oder $S \times \{\Sigma \cup \varepsilon\}$ haben. $\rightarrow P(S)$ (nicht deterministischer Automat), siehe unten.
- s ist der Ausgangszustand, $s \in S$.
- A ist eine Reihe von Empfangszuständen, $A \subseteq S$.



Tätigkeiten an der Maschine

Die Maschine befindet sich zunächst in einem definierten Ausgangszustand. Liest bei jedem Schritt ein Symbol aus dem Eingang und tritt in einen Zustand ein, dem ein Wert zugewiesen wird, der in der Übergabetabelle dem aktuellen Zustand entspricht und das Symbol zurückliest. Es fährt mit dem Lesen des nächsten Symbols des Eingangs, eines weiteren Übergangs durch die Übergangs-Tabelle usw. fort. Je nachdem, ob das Gerät nach dem Lesen des Eingangs in einem Zustand, der zum Satz von Empfängerzuständen gehört, stoppt, akzeptiert der Automat entweder den Eingang oder nicht. Die Menge aller Zeichenketten, die der Automat akzeptiert, bildet eine reguläre Sprache.

12. Turingmaschinen

Church (Church-Touring-)Arbeit: Turingmaschinen (und ihre entsprechenden Systeme) definieren ihre Rechenleistung, die intuitiv als effizient berechenbar angesehen wird. Für jeden Algorithmus gibt es eine gleichwertige Turingmaschine.

Das Church-Argument(Church-Turing) kann nicht formal nachgewiesen werden, wird aber durch eine Reihe von Argumenten gestützt:

Turingmaschinen sind sehr robust - verschiedene Modifikationen verändern ihre Rechenleistung nicht.

Es schlug eine Reihe von verschiedenen Berechnungsmodellen vor, deren Stärke den Turingmaschinen entspricht.

Es gibt keinen bekannten Berechnungsprozess, den wir als effektiv quantifizieren würden und der nicht möglich wäre, mit Hilfe der Turingmaschine zu implementieren.

Die Turingmaschine ist das theoretische Modell eines Computers. Sie besteht aus mehreren Teilen:

- Prozessoreinheit - endlicher Automat
- Programm - Regeln für Übergangsfunktionen
- Rechtsseitiges Endlosband - Wird zur Aufzeichnung von Zwischenergebnissen verwendet.

Es wird für die Modellierung von Algorithmen in der Theorie der Berechenbarkeit verwendet.

Turing's Programmiersprachen und Computer, haben die gleiche Rechenleistung wie Turing's Maschine.

Definition:

$M=(Q,\Gamma,b,\Sigma,s,\delta,F)$	Turingmaschine
Q	Endlicher Satz von internen Zuständen
Γ	Endliches Alphabet von Symbolen auf dem Band
$b\in\Gamma$	Ein leeres Symbol ist nicht Teil des Alphabets der Eingabezeichenkette.
$\Sigma\subseteq\Gamma\setminus b$	Endlicher Satz von Eingangssymbolen

$s \in Q$	Ausgangszustand
$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$	Übergangsfunktion, L nach links nach links, R Kopf nach rechts bewegen
$F \subseteq QQ$	Satz von Endzuständen
$\langle q,s,n \rangle \in Q \times \{yb\hat{}\omega \mid y \in \Gamma\hat{}* \} \times N\_0$	Konfiguration
q	Aktueller Status
s	Der kleinste kontinuierliche Teil des Bandes, der die nicht leeren Symbole enthält.
n	Lesekopfposition (Zellzahl)

Wenn Q,Γ, disjunkte Sätze sind, kann eine kompakte Form verwendet werden.

Band: 1234
Zustand der Maschine: q
Position des Lesekopfes: 2
Konfiguration: 1q234
Erstkonfiguration des TM-Eingangs w∈*
s,wb,0, Kompakte Form: sw

Berechnungsmethode für die Turingmaschine

Wenn der aktuelle Status der Endzustand ist, beendet er die Berechnung.

Der Lesekopf liest ein Symbol aus der Zelle, auf der er sich gerade befindet.

Wenn in der Übergangsfunktion für den aktuellen Zustand ein Übergang definiert ist und ein Übergang für das zu lesende Symbol definiert ist, dann (bei mehreren möglichen Übergängen in nicht-deterministischen Maschinen wird einer zufällig ausgewählt):

Ändert es den Status.
An der aktuellen Kopfposition wird das entsprechende Symbol eingegeben.
Der Kopf bewegt sich entsprechend (verschiebt sich nicht).
Es gibt verschiedene Modifikationen von TM.
TM mit der Möglichkeit, die Berechnung ohne Kopfverschiebung durchzuführen.
Übergangsfunktion um N erweitert - N – $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R,N\}$

Die N-Verschiebung (kein verschieben des Lesekopfes) kann auf einem herkömmlichen TS so angeordnet werden, dass der Lesekopf das Eingangssymbol aus der Zelle liest, auf

der er sich befindet, den entsprechenden Vorgang durchführt und nach rechts fährt (R). Liest nun das Symbol aus der Zelle, schreibt das gleiche Symbol (d.h. ändert nicht das Symbol, auf dem sich der Kopf befindet) und bewegt sich nach links (L). So haben wir den Lesekopf an die vorherige Position gebracht und kein anderes Symbol wurde geändert.

TM mit zweiseitigem Endlosband
Das Band ist nicht linksbündig begrenzt.
Mögliche Verschiebung des Lesekopfes nach links von jeder Konfiguration möglich
N-Band TM
Liest von und schreibt auf mehrere Bänder gleichzeitig.
Die einzige Änderung betrifft die Übergangsfunktion:δ:Q×Γ^n→Q×(Γ×{L,R,N})^n
Nicht-deterministisches TM (NTM)

Ermöglicht "Mehrfachauswahl".

$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R,N\}}$$

Subroutinen werden verwendet, um die Erstellung komplexer TM zu erleichtern. Das Unterprogramm ist der Satz von Zuständen, der den Anfangs- und Endzustand enthält. Normalerweise löst es ein Teilproblem in TM. In der normalen Programmierung ist es das Äquivalent einer Funktion.

Der universelle TM- U als Eingang akzeptiert den Code eines anderen TM T und das Maschineneingangswort T. Er dekodiert die Übergangsfunktion der Maschine T und simuliert die Berechnung der Maschine. Es kann jede partiell rekursive Funktion berechnen (oder entspricht der universellen partiell rekursiven Funktion), entscheidet jede rekursive Sprache und akzeptiert jede rekursive Sprache.

13. Berechenbarkeitstheorie

Die grundlegende Theorie der Frage der Berechenbarkeit lautet: Was ist und was ist nicht algorithmisch berechenbar (lösbar)? Für welche Probleme gibt es Algorithmen, die sie lösen?

Das Problem wird definiert durch:

- Name: XY
- Instanz: Was kann eingegeben werden?
- Ergebnis: Output, Input zu Output Beziehung

Das Problem ist die teilweise Anzeigeart**, gültig für die Eingabe gibt eine gültige Ausgabe für ungültige Eingabe Sonderergebnis "Falsche Eingabe".

Probleme haben die folgenden Eigenschaften:

- Algorithmische Lösbarkeit
  Für ein bestimmtes Problem sagen wir, dass es algorithmisch lösbar ist (oder die jeweilige (Teil-)Ansicht ** ist algorithmisch berechenbar), wenn es einen Algorithmus gibt, der als Eingabe in der Lage ist, jede Instanz des Problems zu akzeptieren und seine Berechnung für einen solchen Eintrag endet immer in der Ausgabe des gewünschten Ergebnisses.

- Algorithmische Entscheidbarkeit
  Über das Problem von Ja / Nein sagen wir, dass der Algorithmus entscheidbar sein kann, wenn es einen Algorithmus gibt, der in der Lage ist, jede Instanz des Problems als Eingabe zu akzeptieren und seine Berechnung für jeden solchen Eintrag endet immer und es wird die Antwort Ja / Nein ausgegeben.

  Das Problem ist algorithmisch lösbar, wenn es einen Algorithmus gibt, der für jedes bestimmte aufgegebene Problem in einer bestimmten Zeit ein Ergebnis eindeutig bestimmt. Entscheidungen für Ja / Nein Fragen.

- Partielle Entscheidbarkeit
  Über das Problem Ja / Nein werden wir sagen, dass das teilweise entscheidbar ist, wenn es einen Algorithmus gibt, dessen Berechnung nur für diejenigen Einträge beendet ist, die mit den Instanzen des Problems mit der Antwort Ja übereinstimmen.

  Das Problem ist teilweise entscheidbar, wenn Sie wissen, dass der Algorithmus stoppt und nur dann ein Ergebnis liefert, wenn die Antwort auf die Frage ja ist. Wenn die Antwort Nein lautet, werden Sie es nie erfahren, denn der Algorithmus hört nie auf.

Die Komplexitätstheorie liefert uns interessante Ergebnisse. Einer von ihnen nennt sich Halting Problem. Wir können keinen Algorithmus konstruieren, der die Endgültigkeit des allgemeinen Programms seines Ablaufs überprüfen würde.

Die Komplexitätstheorie liefert uns eine interessante Hypothese. Einer von ihnen ist die Church-Turing-Thesis. Für jeden Algorithmus gibt es eine gleichwertige Turingmaschine.

- Anhaltendes Problem
  Zuordnung: Wenn Sie den Quellcode des Programms und seine Eingabe kennen, entscheiden Sie, ob das Programm stoppt oder ob es für immer ohne Unterbrechung ausgeführt werden soll.

Im Jahr 1936 bewies Alan Turing, dass es keinen allgemeinen Algorithmus gibt, der das Halteproblem für alle Eingaben aller Programme dort lösen würde. Das Halteproblem wird daher als algorithmisch unentscheidbares Problem bezeichnet.

Beweis:

Die Unentschlossenheit des stoppenden Problems kann durch Widerspruch nachgewiesen werden.

Ursprüngliche Annahme: Nehmen wir an, dass das Halteproblem gelöst werden kann. Das bedeutet, dass es ein Programm stoppt (Programm, Eingabe), welches eine universelle Lösung für das Problem des Stoppens ist - wir gehen daher davon aus, dass, wenn dieses Programm und seine Eingabe übergeben wird, das Programm nach einer endlichen Zeit stoppt, die Antwort so zurückgibt, dass, wenn das aufrufende Programm (Eingabe) nach einer endlichen Anzahl von Schritten vorbei ist, dann stoppt (Programm, Eingabe) wenn JA zurückkehrt, andernfalls (wenn in Managing Calls Programm (Eingabe) in einer Schleife gefangen) NEIN zurückgibt.

Anschließend Konstruieren Sie das Programm Paradox (Programm), das den Aufruf stoppt (Softwareprogramm), und wenn der Aufruf JA zurückkehrt, wissen wir dass er in einer Schleife gefangen ist. Und wenn NEIN zurückgegeben wird, endet er sofort.

Wir fragen, was das Ergebnis eines Aufrufs Paradox (Paradoxon) ist.

Nehmen wir an, dass für einen Moment, der stoppt (Paradox, Paradox), JA ausgegeben wird. Per Definition wissen wir, dass Paradox (Paradox) eine Endlosschleife ist. Per Definition stoppt das Programm, aber wenn die Paradox (Paradox) in einer Schleife gefangen ist, dann muss es stoppen und ein NEIN zurückzugeben. Wir sind also zu einem Widerspruch gekommen.

Andererseits nehmen wir an, dass für einen Moment, der stoppt (Paradox, Paradox), ein NEIN zurückgibt. Per Definition, Programm Paradox dann wissen wir, dass Paradox (Paradox) aufhört. Per Definition stoppt das Programm, aber wenn das Paradoxon (Paradoxon) stoppt, muss es stoppen (Paradoxon, Paradoxon) und JA zurückgeben. Wiederum stoßen wir auf einen Widerspruch.

Da wir alle Möglichkeiten ausgeschöpft haben und immer an Wiedersprüche gelang sind, wird das Problem eine falsche Ausgangsprämisse sein. Folglich existiert das Programm Stopp (Programm, Eingabe), wie definiert, nicht.

Church - Turing-These

Die Hypothese besagt, dass jede mögliche Berechnung erfolgreich durchgeführt werden kann, wenn genügend Zeit und Speicher vorhanden ist.

Der Algorithmus muss die folgenden Anforderungen erfüllen:

- Der Algorithmus besteht aus einer endlichen Anzahl von Anweisungen, die durch die Verwendung einer endlichen Anzahl von Symbolen genau definiert werden.
- Der Algorithmus gibt das Ergebnis immer nach einer endlichen Anzahl von Schritten zurück.
- Den Algorithmus kann sogar einen Mensch mit Bleistift und Papier ausführen.
- Die Ausführung des Algorithmus erfordert keine menschliche Intelligenz, außer dem, was notwendig ist, um Anweisungen zu verstehen und auszuführen.

Da jedes Computerprogramm es in die Sprache einer Turingmaschine übersetzen kann und umgekehrt, kann man eine These in jeder gängigen Programmiersprache formuliert sein.

Die Programmiersprache erfordert eines der folgenden Konstrukte s (unter anderem), dass eine Turing-Komplettlösung (d.h. gleichwertige Turing-Maschine)

- Zyklus während -bis.
- Unbegrenzte (zumindest in der Theorie eklige) Rekursionen,
- Bedingter Sprung.

Gängige Programmiersprachen haben in der Regel alle drei Strukturen. Unter den Sprachen, die komplett sind, ist SQL nicht enthalten (d.h. keine Stored Procedures).

ALGORITHMEN UND DATENSTRUKTUREN

1. Algorithmus

Unter dem Begriff Algorithmus versteht man eine präzise Anweisung oder ein Verfahren, welche bzw. welches in der Lage ist, einen bestimmten Auftrag auszuführen. Er beschreibt das theoretische Lösungsprinzip, anders als ein exakter (spezifischer) Eintrag in einer bestimmten Programmiersprache.

Der Algorithmus sollte bestimmte Eigenschaften haben:

- **Endlichkeit:** Jeder Algorithmus muss nach dem Ausführen der maximal möglichen Anzahl von Schritten enden. Es kann eine beliebige Anzahl von Schritten geben, aber sie muss für jeden individuellen Input begrenzt sein. Ein Verfahren, das sich nicht an diese Regel hält, darf nicht als Algorithmus, sondern nur als Rechenmethode bezeichnet werden.

- **Allgemeingültigkeit:** Der Algorithmus behandelt nicht nur ein spezielles Problem, sondern eine allgemeine Klasse ähnlicher Probleme.

- **Bestimmbarkeit:** Jeder Schritt im Algorithmus muss eindeutig und präzise definiert sein. In jeder Situation muss klar sein, was zu tun ist, wie es zu tun ist und wie der Algorithmus fortsetzen soll. Einige Algorithmen sind nicht vollumfänglich festgelegt: Sie enthalten ein zufälliges Element (zB genetische Algorithmen).

- **Ausgabe (oder Resultativität):** Der Algorithmus hat zumindest eine Ausgabe, die Menge richtet sich nach der vorgeschriebenen Beziehung zu den Eingaben. Die Ausgabe ist die Antwort auf das Problem.

- **Einfachheit:** Der Algorithmus besteht aus einer begrenzten Anzahl einfacher Schritte.

1.1. Entwicklungsverfahren für Algorithmen

Es gibt verschiedene Verfahren, die eingesetzt werden, um Algorithmen zu entwickeln. Dazu zählen unter anderem:

Top-Down-Methode: Prozesslösungen werden in einfachere Operationen zerlegt, bis die elementaren Schritte erreicht sind.

Bottom-Up-Methode: Ausgehend von den elementaren Schritten werden Ressourcen

erstellt, welche es unter Umständen möglich machen, eine spezifische Aufgabe zu bewältigen.

Alternativ gibt es noch eine **Kombination aus beiden Methoden**, wenn eine Top-Down-Methode von einem „partiellen Bottom-Up Schritt" vervollständigt wird (Verwendung der Funktionsbibliothek, höhere Programmiersprache oder Systemprogrammierung.)

Gebräuchliche Verfahren in Algorithmen sind die folgenden Methoden: Teile-und-herrsche, gierige Algorithmen, dynamische Programmierung und Rückverfolgung.

- Die **Teile-und-herrsche Methode** unterteilt die Probleme in Sub-Aufgaben, die unabhängig voneinander sein müssen. Im Anschluss werden diese Sub-Aufgaben gelöst. Diese Methode wird häufig rekursiv oder iterativ angewendet.

- Ein **gieriger Algorithmus (Greedy Algorithm)** wird vor allem eingesetzt, um Optimierungsprobleme zu lösen. Er wählt immer ein lokales Minimum oder Maximum und versucht, davon ausgehend auf ein globales Minimum oder Maximum zu schließen. Dieser Prozess ist nicht immer ideal und führt nicht zwangsläufig zu einem globalen Minimum (Maximum).

- **Dynamische Programmierung** unterteilt das Problem in Sub-Aufgaben, wie es bei der Teile-und-herrsche-Methode der Fall ist. Der grundlegende Unterschied ist, dass es hier zwischen den Sub-Aufgaben Abhängigkeiten geben darf.

- **Die Rückverfolgung (Suchrücklauf)** ist eine Möglichkeit, algorithmische Probleme zu lösen, indem man den Statusbaum des Problems sucht. Es handelt sich hierbei um eine Brute-Force-Suche.

1.2. Arten von Algorithmen

Algorithmen können in verschiedene Arten unterteilt werden:

- **Rekursive Algorithmen**, welche sich selbst verwenden bzw. abrufen.
- **Randomisierte Algorithmen**, welche basierend auf zufälligen (pseudo-zufälligen) Zwischenergebnissen Entscheidungen treffen.
- **Parallele Algorithmen**, welche Aufgaben auf mehrere Computer (Prozessoren, Threads) verteilen.
- Ein weiterer Typ ist der **genetische Algorithmus**, welcher auf der Imitation evolutionsbiologischer Prozesse basiert.
- **Heuristische Algorithmen** suchen nicht nach einer präzisen, spezifischen Lösung, sondern nur nach einem geeigneten Näherungswert. Sie werden in Situationen verwendet, in denen die verfügbaren Ressourcen für die Ausführung exakter Algorithmen nicht ausreichend. Sie kommen auch zum Einsatz, wenn keine pas-

senden exakten Algorithmen bekannt sind.

2. Abstrakter Datentyp – ADT

2.1. Beschreibung

Abstrakter Datentyp – kurz ADT – ist ein Ausdruck für Datentypen, die unabhängig von ihrer eigenen Anwendung sind.

Durch die Verwendung von ADTs versucht man, das Programm, welches Operationen mit dem vorgegebenen Datentyp ausführt, zu vereinfachen und zu straffen.

Jeder ADT kann genutzt werden, wenn man grundlegende algorithmische Operationen wie Belegungen, Additionen, Multiplikationen und bedingte Sprünge verwendet.

2.2. Eigenschaften von ADTs

ADTs sollten die folgenden Eigenschaften haben:

- **Allgemeine Anwendbarkeit:** Einmal entwickelt, können ADTs eingebettet werden und laufen problemlos in jedem Programm.
- **Exakte Beschreibung:** Die Verknüpfung zwischen der Anwendung und der Schnittstelle muss eindeutig und vollständig sein.
- **Einfachheit:** Der Nutzer sollte nicht mit der internen Anwendung und Administration von ADTs im Speicher beschäftigt sein.
- **Abkapselung:** Die Schnittstelle ist ein geschlossener Teil. Der Nutzer weiß, was der ADT macht, aber nicht, wie er funktioniert.
- **Integrität:** Der Nutzer kann nicht in die interne Datenstruktur eingreifen.
- **Modularität:** Das „modulare“ Programmierprinzip ist gut angeordnet und erlaubt den einfachen Austausch von Code-Teilen. Bei der Suche nach Fehlern können die individuellen Module als kompakte Einheiten betrachtet werden. Es ist nicht notwendig, in das gesamte Programm einzugreifen, um ADTs zu verbessern.

Wenn ADTs objektorientiert programmiert sind, werden diese Eigenschaften typischerweise standardmäßig eingehalten.

2.3. Operationen in ADTs

Durch die Verwendung von ADTs können elementare Operationen ausgeführt werden.

Diese umfassen den Konstruktor, Selektor und Modifizierer.

- Der **Konstruktor** erstellt einen neuen ADT. Er bildet – basierend auf bestimmten Parametern – eine gültige interne Darstellung des Werts heraus.

- Der **Selektor** holt die Werte ein, welche die Komponenten oder Eigenschaften eines spezifischen ADT-Werts vervollständigen.

- Der **Modifizierer** nimmt Änderungen an Datentyp-Werten vor.

3. Analyse von Algorithmen

3.1. Effektivitätsvergleich von Prozessen

Ein Problem kann für gewöhnlich durch die Anwendung verschiedener Algorithmen gelöst werden. Es ist daher notwendig, ein Werkzeug zu haben, welches die einzelnen Verfahren hinsichtlich der Effektivität des Prozesses vergleicht. Algorithmen können dabei entweder **experimentell** oder **theoretisch** miteinander verglichen werden.

Eine **experimentelle** Analyse nimmt viel Zeit in Anspruch. Klarerweise dauert sie – je nach Menge und Größe der Eingaben – länger bzw. kürzer. Diese Art der Analyse erfordert eine spezielle Anwendung des Algorithmus, welche wiederum weiteres Wissen voraussetzt. Es ist schwierig, einen durchschnittlichen Fall zu finden. Aus diesem Grund ist es besser, sich auf den schlimmstmöglichen Fall zu konzentrieren. Dieser ist einfach zu analysieren und wichtig für die meisten Anwendungen, seien es nun Spiele, Finanzapplikationen, Robotertechnik oder automatisierte Operationen.

Die experimentelle Analyse der benötigten Zeit findet in einer Umgebung statt, in welcher der Algorithmus (das Programm) meist unter Verwendung einer internen Zeitmessungsfunktion ausgeführt wird. Der Programmdurchlauf hängt von den Eingaben und deren Zusammensetzung ab, und nicht alle Eingaben sind in jedem Programmdurchlauf enthalten. Um zwei Algorithmen vergleichen zu können, benötigt man dieselbe Hard- und Software sowie dieselbe Speicherbelegung.

Anstelle einer experimentellen Analyse können auch bestimmte **theoretische** Verfahren verwendet werden. Die theoretische Analyse verwendet die Beschreibung des Algorithmus mittels Operationen anstatt zur spezifischen Ausführung. Sie berücksichtigt alle Eingaben und ermöglicht es, die Geschwindigkeit des Algorithmus unabhängig von Hardware oder Software einzustufen.

Eines dieser Werkzeuge ist der **Pseudocode**. Dieser erlaubt es, höhere Ebenen von Algorithmus-Beschreibungen zu verwenden. Die Beschreibung ist strukturierter als ein gewöhnlich geschriebener Text, aber nicht so detailliert wie eine spezifische Anwendung. Es handelt sich beim Pseudocode um eine bevorzugte Schreibweise, um Algorithmen zu beschreiben. Sein Vor- und Nachteil gleichermaßen ist, dass er Probleme einer besonderen Anwendung ausblendet. Der Pseudocode verwendet Schlüsselwörter, um den Algorithmus zu beschreiben:

Zur Durchlaufkontrolle:

-If…then…else (condition),
-while…do, repeat…until, for…do (cycles)

Header: Algorithmus `Name(arg1, arg2…)`, input, output

Verfahrensabruf (Methoden, Algorithmus): `var.Name(arg1,arg2,…)`

Wertrückgabe: `return` *Expression*

Ausdrücke:

←	Zuschreibung
=	Gleichheit
+, -, n2,…	Mathematische Operationen

3.2. Primitive Operationen

Eine weitere Möglichkeit ist, Algorithmen mithilfe **primitiver Operationen** zu analysieren.

Eine primitive Operation ist eine grundlegende Operation, welche von einem Algorithmus ausgeführt wird. Sie ist in einem Pseudocode identifizierbar, unabhängig von der Programmiersprache und sollte präzise definiert sein. Solch eine primitive Operation könnte zum Beispiel die Auswertung eines Ausdrucks sein, aber auch die Zuweisung eines Werts zu einer Variable, deren Indexierung in einem Feld, ein Verfahrensabruf, eine Rückgabe usw.

Ebenso kann man eine **asymptotische Notation (big O, Bachmann-Landau Notation)** verwenden. Sie stellt fest, wie operational anspruchsvoll der Algorithmus ist, indem ermittelt wird, wie sich das Verhalten des Algorithmus in Abhängigkeit von der Größe der eingegebenen Daten verändert. Die asymptomatische Zeit und Raumkomplexität kommen hier für gewöhnlich zum Einsatz.

Die verschwendete Schreibweise bedeutet, dass der Algorithmus weniger als A+B f(N) erfordert, wobei A und B adäquat ausgewählte Konstanten sind und N die Variable bezeichnet, welche der Größe der Dateneingabe entspricht. Die zusätzlichen und multiplikativen Konstanten, sprich *O(N+1000)=O(1000 N)=O(N)*, werden ignoriert. Das Interesse gilt rein dem Verhalten eines hohen N-Wertes.

Um die Zeit zu ermitteln, die der Algorithmus, welcher die Big O Notation verwendet, benötigt, muss man die größtmögliche Nummer an primitiven Operationen finden, welche dann mithilfe der Big O Notation ausgedrückt wird.

4. Queues und Stacks

4.1. Stacks

Ein **Stapelspeicher (Stack)** ist eine Datenstruktur, welche dazu dient, Daten zu speichern. Charakteristisch für ihn ist die Möglichkeit zur Datenmanipulation. Er greift auf die Daten zu, indem er das LIFO (**Last In First Out**)-Prinzip verwendet. Man kann sich das al seine Art Container vorstellen.

Der ADT-Stack muss zumindest die folgenden Funktionen enthalten:

- Einfügen eines Objekts
- Rückgabe und Entfernung des letzten Objekts
- Abfragen am oberen Ende des Stacks
- Stack-Größe
- Stack-Inhalt (leer oder Daten vorhanden)

Wenn wir versuchen, eine Pop- oder Top-Operation in einem leeren Stack auszuführen, bekommt man eine *EmptyStackException*-Ausnahme.

Stack-Anwendungen sind beispielsweise der Verlauf des Web-Browsers, die Rückgängig-Reihenfolge in Textverarbeitungsprogrammen oder individuelle Abrufverfahren. Der Stack kann als eine Hilfsdatenstruktur für andere Algorithmen oder als Teil einer anderen Datenstruktur verwendet werden.

Am einfachsten lässt sich ein Stack in Form einer Datenreihe (Array) umsetzen. Elemente werden von links nach rechts ergänzt und die Hilfsvariable besitzt den Index des jeweils letzten Elements.

Dank der Array-Eigenschaften erhält man die folgenden Eigenschaften:

- n – Anzahl der Elemente in dem Stack
- Anforderungen an den Speicher - O (n)
- Dauer der einzelnen Operationen - O (1)

Ein Array unterliegt jedoch auch Einschränkungen:

- Zu Beginn muss die Stack-Größe definiert werden
- Die Stack-Größe kann nicht Wunsch verändert werden
- Das Hinzufügen eines Elements zu einem vollen Stack wird zu einer anwendungsspezifischen Ausnahmebedingung führen

4.2. Queues

Die **Reihen-Datenstruktur (Queue)** setzt auf das FIFO-Prinzip (**First in First Out**).

In ihrer minimalen Umsetzungsform muss eine Queue die folgenden Funktionen enthalten:

- Einfügen eines Objekts am Ende der Queue
- Auswahl eines Objekts vom Anfang der Queue
- Abfrage am Anfang der Queue
- Queue-Länge und –Belegung

Ebenso wie ein Stack kann eine Queue während einer „Aus der Warteschlange enfernen"-Operation eine Ausnahmebedingung ausgeben oder sich über eine leere stack?queue (*EmptyStackException*) einreihen.

Queue-Anwendungen sind zum Beispiel Wartelisten, Warteschlangen, Zugriff auf gemeinsam genutzte Ressourcen wie Drucker oder Multi-Programmierung. Die Queue kann auch als Hilfsdatenstruktur für andere Algorithmen oder als Teil anderer Datenstrukturen verwendet werden.

Die Queue kann mithilfe eines Arrays umgesetzt werden. Um ihre Eigenschaften zu verbessern, wird ein zyklisches Array verwendet. Folglich gibt es zwei Variablen: f, den Index des ersten Elements und r, den Index des letzten Elements plus eins (zeigt auf den ersten freien Platz).

5. Vektoren, Listen und Sequenzen

5.1. Vektoren

Vektoren erweitern das Konzept des Arrays, indem sie die Reihung jedes Objekts speichern. Ein Element im Vektor kann gelesen, eingefügt und entfernt werden, indem man dessen Reihenfolge ermittelt.

Ein Vektor ermöglicht **grundlegende Operationen:**

- Spezifische Reihung von Elementen
- Ersetzen von Elementen an einer bestimmten Stelle
- Einfügen eines Elements an einer bestimmten Position
- Entfernen eines Elements von einer bestimmten Position.
- Darüber hinaus lässt sich feststellen, wie groß ein Vektor ist und ob er leer ist.

Vektorenoperationen können im Fall eines falschen Index' eine Ausnahmebedingung ausgeben, welche in der Regel negativ ist. Eine Vektoranwendung ist eine sortierte Objektsammlung, sprich elementare Datenbank.

Vektoren können mithilfe eines Arrays umgesetzt werden. Dies führt zu folgenden Eigenschaften:

- Variable ***n*** zeigt die Länge des Vektors an
- Operation *isEmpty () elemAtRank (r) replaceAtRank (r, O) - Zeitkomplexität O(1)*
- Operation *insertAtRank (r, O) - Zeitkomplexität O (n)*
- Operation *removeAtRank (R) - Zeitkomplexität O (n)*

5.2. Listen

Eine weitere Datenstruktur ist die Liste. Die Liste ist eine Reihung von Positionen, welche Daten jedweder Form speichert. Sie führt Beziehungen zwischen vor- und nachgereihten Positionen ein.

Allgemeine Operationen sind die Abfrage der Größe und eine leere Listenabfrage. Darüber hinaus kann man herausfinden, ob ein bestimmtes Element das erste oder das letzte ist. Überlässt ist es möglich, das erste und letzte sowie das vorherige und nächste Element zu erhalten.

Die **ADT-Liste** enthält die folgenden **Funktionen**:

Ersetzen	`replaceElement (p,o)`
Vertauschen	`swapElements (p, q)`
Danach einfügen	`insertAfter (p, o)`
Als erstes einfügen	`insertFirst (o)`
Als letztes einfügen	`insertLast (o)`
Entfernen	`remove (p)`

Listen können in eine einfach (Single Linked List) und eine doppelt verketteten Liste (Double Linked List) unterteilt werden. In einer Single Linked List enthält das Element einen Verweis auf den nächsten Knoten. In einer Double Linked List gibt es darüber hinaus einen Verweis auf den vorhergehenden Knoten.

5.3. Sequenzen

Eine ADT-Sequenz ist eine Kombination aus Vektor und Liste, auf Elemente kann daher sowohl durch Verwendung der Position als auch der Reihenfolge zugegriffen werden. Neben den Vektor- und Listenfunktionen enthalten Sequenzen auch die miteinander verbindenden Operationen `atRank (r)` und `rankOf (p)`.

Eine Sequenz ist ein allgemeiner, wesentlicher Datenstrukturtyp der genutzt werden kann, um ein geordnetes Set von Elementen zu speichern. Sie dient als allgemeiner Ersatz für einen Stack, eine Queue, einen Vektor oder eine Liste. Darüber hinaus kann sie als kleine Datenbank verwendet werden.

6. Bäume

Bäume repräsentieren ein hierarchisches Strukturmodell, welches aus Knoten besteht, zwischen denen eine Eltern/Kind-Beziehung besteht. Bäume können als Organigramm, Datensystem oder Programmierumgebung verwendet werden.

Die folgende Terminologie wird verwendet, um Bäume und ihre Teile zu beschreiben.

6.1. Knotenarten

- **Wurzel** (root)
- **Innerer Knoten:** Knoten, der weder eine Wurzel noch ein Blatt ist
- **Liste (Blattknoten, Externer Knoten):** Knoten ohne Spross
- **Elternknoten:** unmittelbar vor einem Knoten befindlicher Knoten auf dem Pfad vom Blatt zur Wurzel
- **Kinderknoten:** unmittelbar auf einen Knoten folgender Knoten auf dem Pfad von der Wurzel zum Blatt
- **Geschwisterknoten:** bezieht sich auf die Knoten mit demselben Elternknoten
- **Vorfahrknoten, Vorgängerknoten:** vor einem bestimmten Knoten liegender Knoten auf dem Pfad zur Wurzel (direkt angrenzender Vorfahrknoten ist der Elternknoten)
- **Nachfolgerknoten:** Knoten, der sich auf dem Pfad von der Wurzel hin zu irgendeinem Blatt hinter einem bestimmten Knoten befindet (nächster Nachfahre ist d. Nachfolger)
- **Tiefe:** Die Baumtiefe ist die Länge des längsten Pfads von der Wurzel zum Blatt, wobei einem leeren Baum der Wert -1 zugewiesen wird
- **Stufe:** wird gemeinhin für eine Sammlung von Knoten verwendet, welche sich in derselben Entfernung zur Wurzel befinden. Gezählt wird die Anzahl der Knoten.

6.2. Struktur

- **Unterbaum:** Sub-Diagramm eines Baums, welches seinerseits ebenfalls ein Baum ist (zumeist werden Unterbäume gebildet, indem man einen Baumknoten als eine neue Wurzel definiert, während der Rest der Struktur beibehalten wird)
- **Ast:** jeder Pfad von der Wurzel zum Blatt

6.3. Funktionen zur Manipulation von Bäumen

Allgemeine Funktionen:

- integer size()
- boolean isEmpty()
- objectIterator elements()
- positionIterator position()

Zugriffsfunktionen:

- position root()
- position parent()
- positionIterator children(p)

Abfragefunktionen:

- boolean isInternal(p)
- boolean isExternal(p)
- boolean isRoot(p)
- Aktualisierungsoperationen
- swapElements(p, q)
- object replaceElement(p, o)

Da ein Baum eine hierarchische Struktur hat, kann er auf verschiedenen Wegen nachverfolgt werden.

Pre-Order Datendurchlauf:

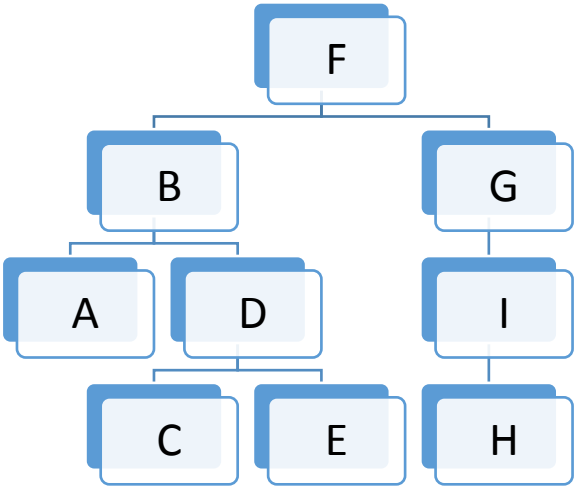
- Überprüfung, ob der Knoten leer oder nicht vorhanden ist
- Anzeigen der aktuellen Knotendaten
- Datendurchlauf des linken Unterbaums mittels eines rekursiven, vertikal hierarchischen Funktionsabrufs (von oben nach unten)
- Datendurchlauf des rechten Unterbaums mittels eines rekursiven, vertikal hierarchischen Funktionsabrufs (von oben nach unten)

In-Order Datendurchlauf:

- Überprüfung, ob der Knoten leer oder nicht vorhanden ist
- Anzeigen der aktuellen Knotendaten
- Datendurchlauf des linken Unterbaums mittels eines rekursiven, aufsteigend sortierten Funktionsabrufs (chronologisch/alphabetisch)
- Datendurchlauf des rechten Unterbaums mittels eines rekursiven, aufsteigend sortierten Funktionsabrufs (chronologisch/alphabetisch)

Post-Order Datendurchlauf:

- Überprüfung, ob der Knoten leer oder nicht vorhanden ist
- Anzeigen der aktuellen Knotendaten
- Datendruchlauf des linken Unterbaums mittels eines rekursiven, vertikal hierarchischen Funktionsabrufs (von unten nach oben)
- Datendurchlauf des rechten Unterbaums mittels eines rekursiven, vertikal hierarchischen Funktionsabrufs (von unten nach oben)



Jede Art des Datendurchlaufs führt zu einem anderen Resultat

- **Pre-Order-Ergebnis:** F, B, A, D, C, E, G, I, H
- **In-Order-Ergebnis:** A, B, C, D, E, F, G, H, I
- **Post-Order-Ergebnis:** A, C, E, D, B, H, I, G, F

Wenn man einen ADT-Baum verwendet, ist es des Weiteren möglich, einen Binär-Baum oder andere Arten zu definieren.

Der Binär-Baum erweitert die Definition des Baums um die Vorgabe, dass kein Knoten mehr als zwei Kinder haben darf, womit ein geordnetes Paar aus linkem und echtem Nachfahren gebildet wird.

Der Binär-Baum fügt zusätzliche Funktionen hinzu:

- position leftChild(p)
- position rightChild(p)
- position sibling(p)

7. Vorrang-Queues und Heaps

7.1. Vorrang-Queues

Eine **Vorrang-Queue** (Vorrangwarteschleife) speichert eine Objektsammlung, in der die Objekte nach Schlüssel/Wert-Paaren (Vorrangigkeit) geordnet sind.

Deren **elementare Umsetzung** sollte folgende Funktionen enthalten:

- InsertItem (k, o)
- removeMin ()
- Minkey (k, o)
- minElement ()
- Size ()
- isEmpty ()

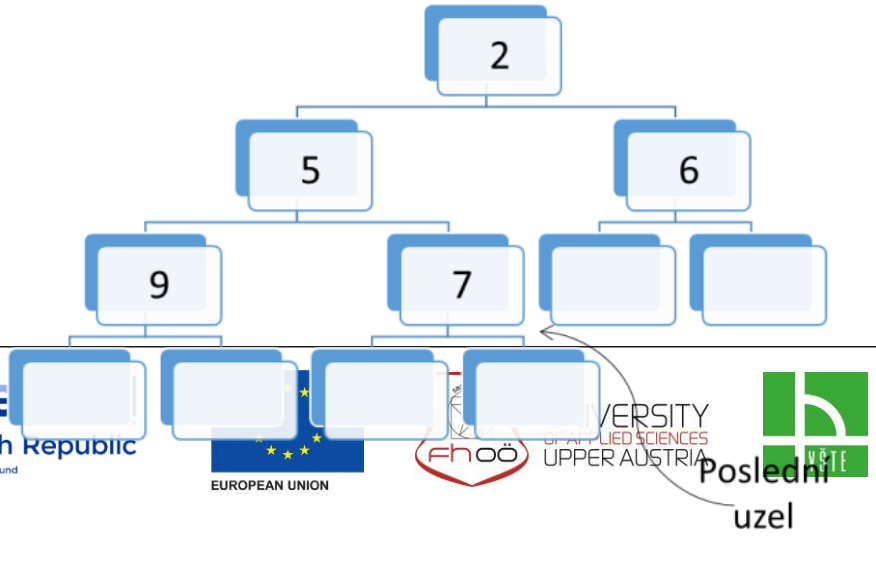
Beispiele für Vorrang-Queues sind Auktionen und Aktienbörsen.

Der **Vorrang-Queue-Schlüssel** kann jedes Objekt mit einer definierten Reihenfolge sein, welches sich ordnen lässt. Zwei verschiedene Elemente (Werte) können denselben Schlüssel (Vorrang) haben. Um eine Vorrang-Queue zu verwenden, muss eine ADT-Vergleichseinrichtung implementiert werden, welche den Vergleich zweier Objekte ermöglicht.

7.2. Heaps

Bei einem **Heap** handelt es sich um einen Binär-Baum, welcher Schlüssel als interne Knoten abspeichert. Für jeden Knoten außerhalb der Wurzel gilt, dass dessen Knotenschlüssel relevanter ist als jener des Elternknotens. Für jeden Heap wird ein kompletter Binär-Baum definiert. Wenn h die Höhe des Baums ist, gibt es für *i* von *0* bis *h-1 2i* Knoten mit der Tiefe *i*.

Der letzte Heap-Knoten ist ein interner Knoten, welcher sich am äußersten rechten Rand auf der Stufe *h -1* befindet.



Letzter
Knoten

Ein Heap kann eingesetzt werden, um eine Vorrang-Queue zu realisieren. Im Anschluss wird ein Objekt (Schlüssel, Wert) in jedem internen Knoten gespeichert, wobei ein Verweis auf die Position des letzten Elements behalten wird.

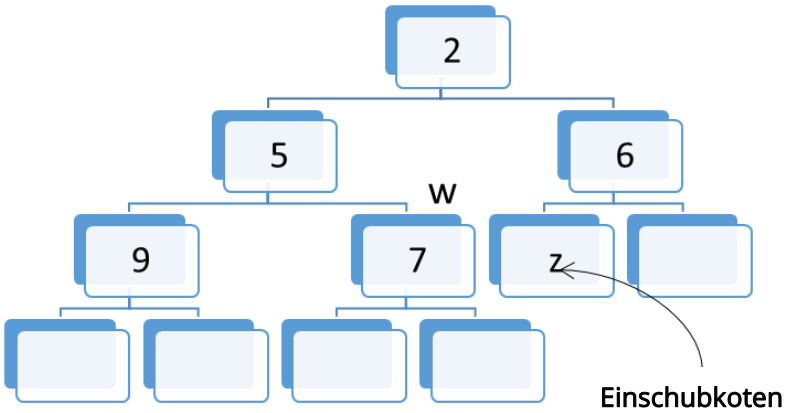
7.3. Handhabung von Heaps

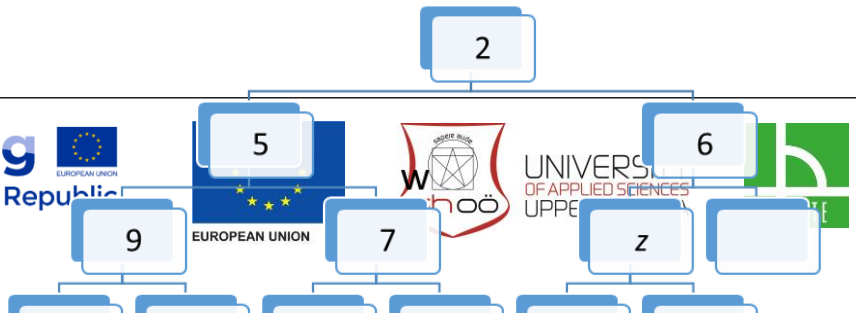
InsertItem (k, o) Funktion

- Aufspüren des Knotens, in den das Objekt eingefügt wird
- Speichern des k-Schlüssels in den z-Knoten, der z-Knoten wird in einen internen Knoten umgewandelt
- Wiederherstellung der Heap-Reihenfolge (Prüfung der Eigenschaften) – upheap Funktion

RemoveMinch () Funktion

- Bezieht sich auf die Entfernung der Wurzel vom Heap (Knoten 2)
- Verändert die Wurzel für den letzten Knoten (2-7)
- Verändert den w-Knoten and seine Kinder in der Liste
- Stellt die Heap-Reihenfolge wieder her – downheap () Funktion





upheap()

- Das Einfügen eines Knotens kann sich auf die Anordnung/das Layout auswirken
- Der upheap-Algorithmus stellt die Ordnung wieder her, indem er den k-Schlüssel oberhalb des eingefügten Knotens austauscht
- Er endet, wenn der interne Knoten zur Wurzel wird oder der Elternschlüssel kleiner oder gleich k ist

downheap()

- Das Entfernen eines Knoten kann sich auf die Anordnung/das Layout auswirken
- Der downheap-Algorithmus stellt die Ordnung wieder her, indem er den k-Schlüssel unterhalb des entfernten Knotens austauscht
- Er endet, wenn der interne Knoten zu einem Blatt wird oder der Kind-Schlüssel größer oder gleich k ist

8. Dictionary und Hashtabellen

8.1. Dictionary

Ein **Dictonary** bezieht sich auf einen ADT, welcher eine Schlüssel/Wert-Sammlung enthält, nach der gesucht werden kann. Die folgenden **Funktionen** können mithilfe eines Wörterbuchs durchgeführt werden:

- FindElement (k)
- insertItem (k, o)
- removeElement (k)
- size ()
- isEmpty ()
- keys ()

Beispiele für Wörterbuchanwendungen sind Verzeichnisse, die Kreditkartenautorisierung, Wörterbucher sowie die Domainnamen-Übersetzung in eine IP-Adresse.

Ein Logdaten-Wörterbuch wird in Form einer Zufallssequenz als Double Linked List umgesetzt. Die Objekte werden dabei in einer beliebigen Reihenfolge gespeichert.

Komplexität der Funktionen:

- Objekt einfügen O (1)
- Element finden, Element entfernen O (n)

Eine **Log-Datei** ist für kleine Wörterbücher oder Anwendungen geeignet, in welchen die am häufigsten gebrauchte Funktion das Einfügen ist, jedoch nur selten etwas gesucht oder entfernt wird.

Die Funktion **findElement (k)**, welche in Form einer Sequenz in ein Wörterbuch implementiert ist, baut auf einem Schlüssel-basierten Array auf und wird als binäre Suche ausgeführt. Bei jedem Schritt wird die Anzahl der Kandidaten durch zwei geteilt, wobei die Funktion nach einer logarithmischen Anzahl an Schritten endet.

Eine **Nachschlagetabelle** ist ein Wörterbuch, welches mithilfe einer geordneten Sequenz umgesetzt wird. Hier werden die Wörterbuch-Einträge in einer Sequence gespeichert, welche auf einem Schlüssel-basierten Array aufbauen. In diesem Zusammenhang wird ein externer Schlüssel-Komparator benötigt.

Komplexität der Funktionen:

- Aufspüren des Elements *O (log (n))*
- Einfügen eines Elements, Entfernen eines Elements *O (n)*

Eine Nachschlagetabelle erweist sich als effektiv bei kleinen Wörterbüchern oder Anwendungen, wo Suchen eine häufig verwendete Funktion ist.

Der **Binäre Such-Baum** ist ein binärer Baum, für welchen die folgenden Regeln gelten:

- u, v und w sind solche Knoten, für welche es stimmt, dass sich u in dem linken Unterbaum von v und dass sich w im rechten Unterbaum von v befindet
- $key(u) \leq key(v) \leq key(w)$
- Externe Knoten speichern keinerlei Objekte
- In-Order-Datendurchlauf vergibt die Schlüssel in aufsteigender Reihenfolge

8.2. Hashtabelle

Hash-Funktion *h* ist eine Funktion, welche einem Schlüssel einer bestimmten Art einen Integralwert aus dem Intervall zwischen *0* und *N-1* zuweist. Der Zweck dieser Funktion ist die gleichmäßige Aufteilung der Schlüssel bei einem bestimmten Intervall.

Eine Hashtabelle für eine bestimmte Schlüsselart enthält eine Hash-Funktion und ein Array (eine Tabelle) der Größe N.

Der Schlüssel wird durch einen Hash-Wert ersetzt. Es kann jedoch auch vorkommen, dass für die zwei Schlüssel derselbe Hash-Wert generiert wird. In diesem Fall tritt eine Kollision auf. Dieses Problem kann auf zwei Arten gelöst werden:

- Variante 1 ist die **Verkettung**, in deren Zuge die miteinander in Konflikt stehenden Objekte als Sequenzen abgespeichert werden.

- Eine weitere Möglichkeit ist das **freie Adressieren**, bei dem das Objekt an einer anderen Stelle in der Tabelle gespeichert wird.

9. Sortieralgorithmen

9.1. Erklärung

Sortieralgorithmen werden eingesetzt, um die Daten in einer Datei in einer speziellen Reihenfolge zu sortieren, entweder alphabetisch oder numerisch. Das Schlüssel/Wert-Paar wird nach dem Schlüssel gewichtet, der Wert findet hingegen keine Berücksichtigung.

Sortieralgorithmen können in **stabile** und **instabile** unterteilt werden, was davon abhängt, ob sie die Reihenfolge der Objekte mit demselben Schlüssel beibehalten. Unterschieden werden kann auch zwischen **natürlichen** und **unnatürlichen** Sortieralgorithmen, wobei natürliche mit einem teilweise geordneten Datensatz schneller arbeiten.

Darüber hinaus können sie auf den **Sortiertyp** heruntergebrochen werden:

- durch Auswahl
- durch Einfügen
- durch Ersetzen
- durch Zusammenführen

Die **bekanntesten Algorithmen** sind

- Bubble Sort
- Heap Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Selection Sort

Andere Algorithmen, welche auf verschiedenen Prinzipen basieren, sind unter anderem:

- Bucket Sort
- Radix Sort
- Counting Sort

9.2. Bubble Sort

- Einfach zu implementieren
- Universell, lokal (in-situ, kein Bedarf an zusätzlichem Speicherplatz)
- Der Algorithmus beginnt am Anfang des Datensets. Er vergleicht die ersten zwei Elemente und wenn das erste größer als das zweite ist, vertauscht er diese. Diesen Vorgang wiederholt der Algorithmus bei jedem Paar benachbarter Elemente, bis er am Ende des Datensatzes angekommen ist. Danach beginnt der von Neuem mit den ersten zwei Elementen und wiederholt diesen Vorgang so lange, bis im letzten Durchlauf kein Austausch mehr möglich ist.

Der Bubble Sort-Algorithmus im Detail:

```
procedure bubbleSort( A : list sortable items )
    n = length(A)
    repeat
        swapped = false
        for i from 1 to n-1 inclusive do
     if A[i-1] > A[i] then
                        swap( A[i-1], A[i] )
                swapped = true
            end if
        end for
    until not swapped
end procedure
```

9.3. Heap Sort

- Ein vergleichsbasierter Sortieralgorithmus
- Nicht stabil
- Verwendet die Heap-Datenstruktur und deren Eigenschaften

Der Heap Sort-Algorithmus im Detail:

```
procedure heapsort(a, count) is
    input: an unordered array a of length count
    heapify(a, count)
   end ← count - 1
    while end > 0 do
         swap(a[end], a[0])
        (the heap size is reduced by one)
        end ← end - 1
        (the swap ruined the heap property, so restore it)
        shiftDown(a, 0, end)
```

Erstellen eines Heaps aus einem Array: Wenn das kleinste Element die Wurzel ist, wird es an die erste Stelle im Array gesetzt und die Wurzel wird entfernt.

Downheap(): Wiederherstellen des Heaps durch Befolgen der Regeln: Die Wurzelentfernung und Heap-Wiederherstellung wird so lange wiederholt, bis der Heap leer ist.

9.4. Insertion Sort

- Ein einfacher Sortieralgorithmus, welcher der unsortierten Eingabefolge ein beliebiges Element entnimmt und dieses an der richtigen Stelle in die zu Anfang noch leere Ausgabefolge einfügt
- Einfache Implementation
- Effizient bei (ziemlich) kleinen Datensätzen
- Effizient bei Datensätzen, die bereits zum Teil sortiert sind
- Stabil, on-line, in-place

Der Insertion Sort-Algorithmus im Detail:

```
for i = 1 to length(A)
    j ← i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j ← j - 1
    end while
end for
```

9.5. Merge Sort

- Ist ein effizienter, allgemein-zweckmäßiger, vergleichsbasierter Sortieralgorithmus
- Verwendet die Teile-und-herrsche-Methode
- Ein stabiler Teile-und-herrsche-Algorithmus, leicht zu parallelisieren
- Grenz- und durchschnittliche Zeit: *O(NlogN)*
- Bedarf an zusätzlichem Speicher: ein Array der Größe N

Der Merge Sort-Algorithmus im Detail:

```
mergesort(m)
      var list left, right
      if length(m) ≤ 1
            return m
      else
            middle = length(m) / 2
            for each x in m up to middle
                  add x to left
            for each x in m after middle
                  add x to right
left = mergesort(left)
            right = mergesort(right)
            result = merge(left, right)
      return result
```

9.6. Quick Sort

- Effizienter Sortieralgorithmus
- Benötigt: O(NlogN) – O(N2)
- Teile-und-herrsche-Algorithmus, nicht stabil, in-place
- Rekursiv

- Verwendet die folgenden Schritte:
    - Auswahl eines Dreh- und Angelpunkts aus dem Array.
    - Partitionierung: Das Array wird neu geordnet, sodass alle Elemente mit einem Wert, der geringer ist als jener des Angelpunkts, vor ebendiesem gereiht werden. Elemente mit einem höheren Wert werden danach eingeordnet, identische Werte können sowohl vor als auch nach dem Angelpunkt gereiht werden. Nach dieser Partitionierung ist der Dreh- und Angelpunkt an seiner finalen Stelle. Dieser Vorgang nennt sich Partitionsoperation.
    - Die oben genannten Schritte sollen in der Folge auch rekursiv für die beiden Sub-Arrays angewendet werden, welche die Elemente mit den kleineren und größeren Werten enthalten. Hierbei ist darauf zu achten, dass die Schritte in jedem Sub-Array separat angewendet werden.

- Auswahl des **Dreh- und Angelpunkts**: der Median ist ideal
    - Erstes Element (beliebig festgelegte Position) – nicht geeignet für bereits teilweise vorsortierte Datensätze
    - Zufälliges Element – in Wahrheit pseudo-zufällig
    - Median aus drei (fünf…) – oder einer anderen Anzahl an Elementen, die sich an den festgelegten oder zufälligen Positionen befinden

Wenn der Dreh- und Angelpunkt richtig gewählt ist, bedarf es keines zusätzlichen Speicherplatzes. Quicksort ist ein instabiler Algorithmus. Die Methode der Angelpunkt-Auswahl beeinflusst die Sortierung, aber im Schnitt ist es der schnellste allgemeine Sortieralgorithmus für Arrays im operationalen Computerspeicher.

9.7. Selection Sort

Ein einfacher Algorithmus, dessen Zeitkomplexität O (N 2),beträgt. Es ist für kleine Datenvolumina geeignet. Er ist allgemeingültig, lokal und instabil.

Ablauf:

- Unterteilen der Sequenz in einen geordneten und ungeordneten, unbestimmten Teil
- Aufspüren des Elements mit dem geringsten Wert in dem ungeordneten Teil dieser Sequenz

- Ersetzen des Elements durch das Element an der ersten Stelle des ungeordneten Teils
- Das erste Element des ungeordneten Teils wird in den geordneten Teil aufgenommen. Gleichzeitig wird der ungeordnete Teil um ein Element von links reduziert.
- Der Restbestand der Sequenz wird durch das Wiederholen der Schritte 2 bis 5 im verbleibenden ungeordneten Teil geordnet.

Überblick: Vergleich der Sortieralgorithmen

Name	Time complexity			Extra memory	Stable	Natural	Method
	Minimum	Average	Maximum				
Bubble sort	*O(n)*	$O(n^2)$	$O(n^2)$	*O*(1)	yes	yes	Exchange
Heapsort	*O*(*n* log *n*)	*O*(*n* log *n*)	*O*(*n* log *n*)	*O*(1)	no	no	Heap, exchange
Insertion sort	*O(n)*	$O(n^2)$	$O(n^2)$	*O*(1)	yes	yes	Inserting
Merge sort	*O*(*n* log *n*)	*O*(*n* log *n*)	*O*(*n* log *n*)	*O*(log n)	yes	yes	Merging
Quicksort	*O*(*n* log *n*)	*O*(*n* log *n*)	$O(n^2)$	*O*(log *n*)	no	no	Exchanging
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	*O*(1)	no	no	Selection

9.8. Bucket Sort

Dieser Sortieralgorithmus unterteilt Daten in verschiedene Eimer, die dafür benötigte Zeit ist O (n * k), wobei k = n / m, Eingabe n, die Anzahl der Eimer ist m.

Um Bucket Sort verwenden zu können, sind **bestimmte Bedingungen** erforderlich:

- Geeignet für gleichmäßig verteilte Eingabedatenwerte
- Der Algorithmus zur Anordnung der Eimer muss stabil sein

Verfahren:

- Die Eingabedaten werden auf eine vordefinierte Anzahl an Eimern verteilt.
- Für jeden Eimer wird ein stabiler Sortieralgorithmus abgerufen.
- Die individuellen Eimer werden schrittweise in das Ausgabe-Array kopiert.

Die **Vorteile** von Bucket Sort sind, dass der Algorithmus gut parallelisierbar ist und dass

er nicht alle Daten gleichzeitig im Speicher benötigt.

9.9. Radix Sort

Der Algorithmus sortiert ganze Zahlen, indem er alle Ziffern durchsucht. Es gibt zwei Ansätze:

- LSD (Least Significant Digit)
- MSD (Most Significant Digit)

Benötigte Zeit: *O((z+n)*logzu)*, wobei z die Basis des ausgewählten Zahlensystem ist, n für die Zahlen bei der Eingabe steht und u dem Maximum an Zahlen bei der Eingabe entspricht.

Radix Sort eignet sich nicht für eine unbegrenzte Eingabegröße.

Beispiel eines LSD-Radix: 170, 45, 75, 90, 802, 2, 24, 66 ⇒ 170, 90, 802, 2, 24, 45, 75, 66 ⇒ 802, 2, 24, 45, 66, 170, 75, 90 ⇒2, 24, 45, 66, 75, 90, 170, 802

9.10. Counting sort

Stabil und geeignet für große Daten mit einer kleinen Menge abstrakter Werte. Die benötigte Zeit ist *O(N+M)* und der zusätzliche Speicherbedarf beträgt *O(M.)*

Voraussetzungen:

- Die Anzahl der verschiedenen Werte (M) ist erheblich geringer als die Gesamtzahl der Elemente (N).
- Hilfs-Array: Es schreibt und liest zeitlich konstant (das Array wird durch Werte oder Hash-Werte indexiert)

Spezifika des Algorithmus:

- Er durchläuft das Eingabe-Array von links (oder von rechts)
- Jedes Element tritt häufiger im Hilfs-Array auf
- Jedem Objekt wird die Zahl des Auftretens aller vorhergehenden Objekte (der Algorithmus erhält die exakte Position der Grenze) beigefügt
- Beginnt, das ungeordnete Array von rechts zu durchlaufen
- Bei jedem Element schaut der Algorithmus in das Hilfs-Array an der Spitze der Anordnungsgrenze
- Counting Sort platziert das Element in dieser Grenze und reduziert es um eins
- Dieser Vorgang wird wiederholt, bis das gesamte Array durchlaufen ist

10. Mustererkennung und Präfixbäume

10.1. Mustererkennung

Mustererkennung ist im Wesentlichen die Suche einem bestimmten Muster in einer vorgegebenen Sequenz. Für gewöhnlich wird nach einer Sub-Zeichenfolge in einer Zeichenkette (String) gesucht.

Zunächst muss hierfür ein String defniert werden. Ein String ist eine Sequenz aus Zeichen eines vorgegebenen Alphabets. Ein Alphabet ist ein Sammlung aller möglichen Zeichen, zB ASCII, UniCode, {0,1} oder {A, C, G, T}.

Wenn die Länge des *P*-Strings *m* beträgt, dann besteht der Sub-String *P [i..j]* von P aus Zeichen zwischen *i* und *j*. Der String vor dem Index *i* ist ein Präfix. Der String, welcher sich hinter dem Index *j* befindet, ist ein Suffix.
Anwendungsgebiete: Textverarbeitungsprogramme, Suchwerkzeuge, Biologische Forschung

Es gibt einige Algorithmen für die Mustererkennung:

Der wohl wesentlichste ist der **Brute-Force-Algorithmus**.

Dieser durchläuft den Text von links nach rechts.

Er vergleicht das P-Muster an jeder möglichen Stelle mit dem T-Text bis:

- Eine Übereinstimmung gefunden wird
- Alle möglichen Positionen überprüft wurden

Die Zeit, die dieser Algorithmus benötigt, beträgt *O (nm).*

Der **Boyer-Moore Algorithmus** beginnt den Text vom Ende an zu durchlaufen, sprich von rechts nach links.

Hier gilt es folgendes zu definieren: Der Index *i* zeigt die Stelle im Text *T* an, der Index *j* zeigt auf *P*.

Während der Suche können vier mögliche Situationen auftreten:

- *T (i)* ist nicht in *P* enthalten: In diesem Fall wird i an der Länge von *P* vorbeigeschoben (*P* wird auf den nächsten Buchstaben T ausgerichtet, welcher *T (i + 1)* ist)
- *T (i)* stimmt mit *P (j)* überein – Es werden beide nach links versetzt und der Vor-

gang wird wiederholt (wie im Brute-Force-Algorithmus)
- *T (i)* ist *P (j), T (i)* steht vor dem Index *j P -> P* wird nach rechts ausgerichtet, damit *T (i)* mit dem Auftreten von *P* übereinstimmt und der Vorgang wird wiederholt.
- *T (i)* ist *P (j), T (i) P* ist ein Index *j -> P* wird um eins nach rechts ausgerichtet und der Vorgang wird wiederholt (nicht wiederkehrend, aber nicht weiter nach unten versetzend)

Rückgabe von *i*, wenn das gesamte Muster gefunden wird.

Dieser Algorithmus ist schneller als der Brute-Force-Algorithmus, allerdings kann dessen Komplexität *O (mn + A)* betragen, wobei A der Größe des Alphabets entspricht.

Um diesen Algorithmus anwenden zu können, ist eine Datenvorbehandlung notwendig.

- In deren Zuge wird die Position der Buchstaben, beginnend von links, herausgefunden
- Sieht das Muster beispielsweise wie folgt aus: "ABRAKADABRA",
- So erhält A den Index 10, *B* erhält den Index 8, *K* = 4, *D* = 6 und *R* = 9. Anderen Buchstaben wird der Wert -1 zugewiesen.
- Im Anschluss wird die Funktion Last (char input) implementiert, welche den Index gemäß den Buchstaben zurückgibt. Dann, in den Fällen 3 und 4, werden der letzte *(T (i))* und *j* verglichen, woraus sich schließen lässt, ob der Algorithmus an die Stelle Last *(T (ii)* (wenn Last *(T (i))* <j ist) oder nur *i++* versetzt werden soll.

Knuth-Morris-Pratt (KMP) Algorithmus

Dieser Algorithmus sucht den Text von links nach rechts und unterscheidet sich dadurch vom Brute-Force-Algorithmus, welcher nicht alle Vergleiche durchführt. Wenn eine Unstimmigkeit gefunden wird, verschiebt sich der Algorithmus um mehr als einen Buchstaben. Wenn ein *P* (vom Anfang, ein Präfix) gefunden wird, stimmen die Zeichen dieses Präfix‘ mit dem Text überein, weshalb keine Notwendigkeit für eine erneute Prüfung besteht. Das Ende des gefundenen Sub-Strings kann auch am Anfang dieses Sub-Strings enthalten sein. Solch eine Übereinstimmung entspricht klarerweise dem ganzen gefundenen *P*, weshalb nach einem *P + 1* gesucht wird. Deshalb geht man vom linken und rechten Ende des gefundenen *P*-Stücks aus, und wenn man keine Übereinstimmung findet, wird klar, wie viel man verschieben kann. Dies kann in einer Tabelle gerechnet werden: In diesem Fall ist alles *O(1).*

10.2. Präfixbaum

Ein Präfixbaum ist eine Baumstruktur, die dazu dient, einen Text vorzubereiten. Jeder Knoten hat einen Buchstaben. Die Länge des Pfads vom Knoten zur Spitze stellt die Posi-

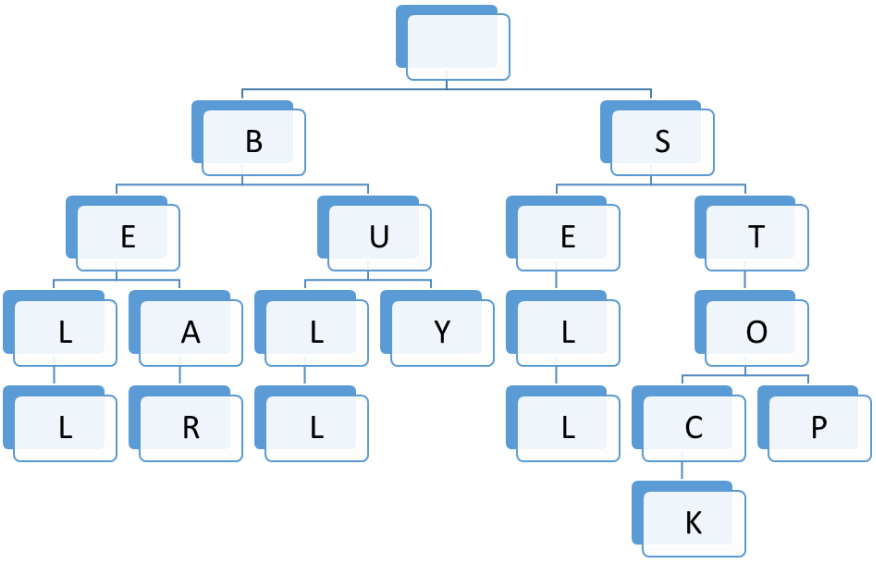
tion des Buchstabens im Wort fest.

Die Suche erfordert eine bestimmte Zeit *O (dm),* wobei d der Größe des Alphabets und m der Länge des Worts entspricht.

Es ist möglich, den gesamten Text in einer Präfixbaum-Struktur zu speichern. Nach diesem Vorgang enthält jeder Knoten ein Wort.

Zum Speichern kann ein sogenannter komprimierter Präfixbaum definiert warden. Dieser Baum enthält dann zumindest Knoten zweiten Grades (zwei Buchstaben pro Knoten).

Beispiel: S={BELL, BEAR, BULL, BUY, SELL, STOCK, STOP}

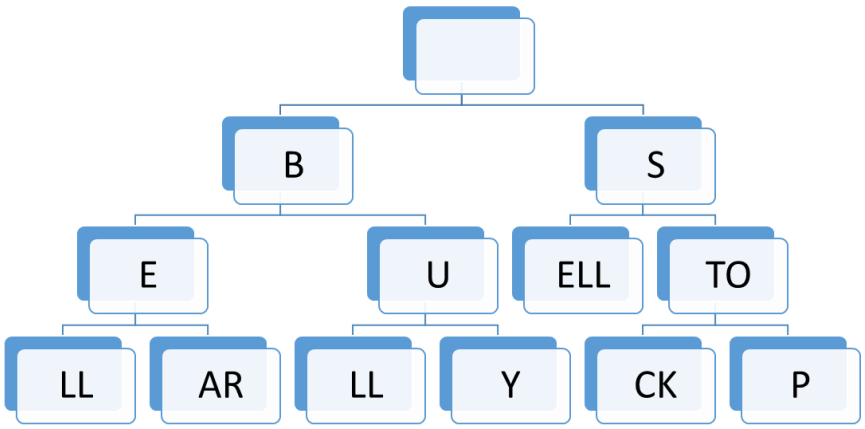


11. Graphentheorie

Ein Graph ist ein geordnetes Paar *(V, E)*, wobei *V* einem Set aus Scheitelpunkten und E einem Set von Kanten entspricht. Jede Kante wird nur mittels zweier Scheitelpunkte bestimmt, oder optional durch die Richtung oder das Gewicht.

11.1. Arten von Kanten:

- **Orientierte**: geordnetes Paar von Scheitelpunkten *(u, v)*, wobei u der Anfang und v das Ende bzw. das Ziel ist
- **Orientierungslose:** geordnetes Paar von Scheitelpunkten *(u, v)*



- **Schleifen:** Kante beginnt und endet am selben Scheitelpunkt
- **Multiple Kanten (mehrfach, parallel):** mehrere Kanten zwischen den Scheitelpunkten *(u, v)*

11.2. Arten von Graphen

- **Orientierte:** Alle Kanten sind ausgerichtet
- **Orientierungslose:** Keine Kante ist ausgerichtet
- **Multigraph:** enthält mehrere Kanten

11.3. Terminologie

- Endscheitelpunkte (oder Endpunkte) einer Kante
- Kanten-Störfall an einem Scheitelpunkt
- Benachbarte Scheitelpunkte
- Grad eines Scheitelpunkts
- Parallele Kanten
- Eigen-Schleife
- Pfad
    - Sequenz alternierender Scheitelpunkte und Kanten
    - Beginnt mit einem Scheitelpunkt
    - Endet mit einem Scheitelpunkt
    - Direkt vor und nach jeder Kante sind deren Endpunkte
- Einfacher Pfad
    - Ein Pfad, bei dem sich alle Scheitelpunkte und Kanten voneinander unterscheiden
- Zyklus
    - Zirkuläre Sequenz alternierender Scheitelpunkte und Kanten
    - Direkt vor und nach jeder Kante sind deren Endpunkte
- Einfacher Zyklus
    - Ein Zyklus, bei dem sich alle Scheitelpunkten und Kanten voneinander unterscheiden

11.4. Graph-ADT-Funktionen

Zugangsfunktionen

- aVertex()
- incidentEdges(v)
- endVertices(e)
- isDirected(e)
- origin(e)
- destination(e)
- opposite(v,e)
- areAdjecent(v,w)

Update-Funktionen

- insertVertex(o)
- insertEdge(v, w, o)
- insertDirectedEdge(v, w, o)
- removeVertex(v)
- removeEdge(e)

Allgemeine Funktionen

- numVertices()
- numEdges()

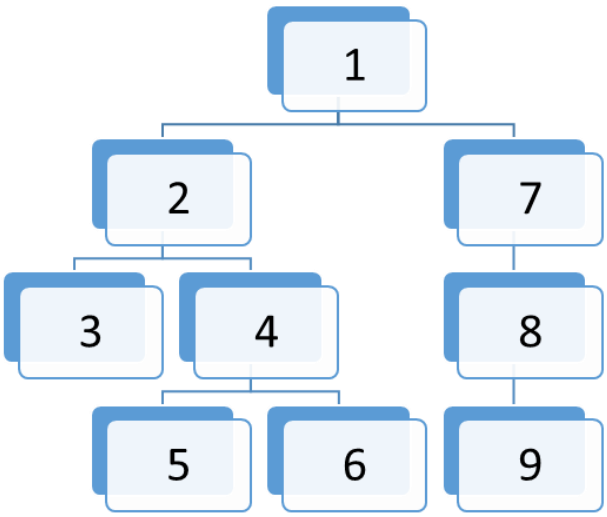
- vertices()
- edges()

Die Graphen-Struktur muss irgendwie durchsucht werden. Hierfür gibt es zwei Optionen: die DFS (Depth-First Search) und die BFS (Breadth-First Search).

11.5. Depth-First Search

Die **Depth-First Search** ist ein vollständiger Algorithmus, welcher jeden Knoten durchläuft. Sein Prinzip besteht im Umstand, dass er den ersten Nachfolger jedes Höchstwerts vergrößert, wenn dieser noch nicht besucht wurde. Wenn er auf einen Höchstwert stößt, von dem aus es nicht möglich ist den Vorgang fortzusetzen (es gibt keine Nachfolger oder alle davon sind bereits besucht worden), geht er mittels Rückverfolgung zurück.

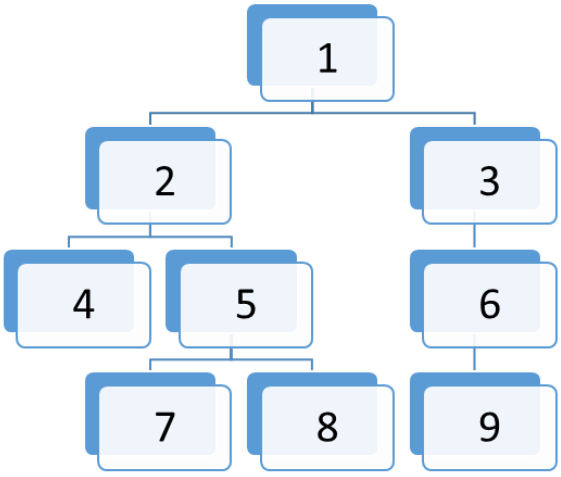
Er durchläuft die Knoten in der folgenden Reihenfolge:



11.6. Breadth-First Search

Breadth-First Search ist ein ähnlicher Algorithmus, der alle Nachbarn ausgehend vom anfänglichen Höchstwert durchläuft, danach die Nachbarn der Nachbarn usw., bis die gesamte Verbindungskomponente durchlaufen ist.

Er durchläuft die Knoten in der folgenden Reihenfolge:



Eine essentielle Frage der Graphen-Theorie ist es, wie man den kürzesten Pfad zwischen zwei Knoten findet. Hierfür gibt es verschiedene Algorithmen.

Dijkstras Algorithmus

- Keine negativen Kantenwerte
- *O(|V|2+|E|)* – *V* Anzahl der Scheitelpunkte, *E* Anzahl der Kanten

Bellman-Ford Algorithmus

- Graph kann negative Kanten haben
- *O(V E)* – langsamer als Dijsktras Algorithmus

Floyd-Warshall Algorithmus

- Ausgerichteter/orientierter Graph ohne negative Kanten
- Findet den kürzesten Pfad zwischen allen Scheitelpunkten
- Benötigte Zeit: *O(V3)*, benötigter Speicher: *O(V²)*

Johnsons Algorithmus

- Orientierter Graph, kann negative Kanten enthalten
- Findet den kürzesten Pfad zwischen allen Scheitelpunkt-Paaren in einem zerstreuten, nach Kanten gewichteten, ausgerichteten Graphen.
- Erlaubt, dass einige Kantenwerte negative Zahlen sind
- O(V2 log2(V)+ VE)

12. Genetische Algorithmen

12.1. Erklärung

Genetische Algorithmen gehören zu den evolutionären Algorithmen und sind Teil der künstlichen Intelligenz. Sie sind eine Klasse der heuristischen Algorithmen. Sie nutzen das Wissen der Evolutionsbiologie, um komplexe Probleme zu lösen, für die es keine exakten Algorithmen gibt. Sie imitieren dazu die Methoden der Evolutionsbiologie:

- Vererbung
- Mutation
- Natürliche Auslese
- Kreuzung

Genetischer Algorithmen arbeitet nach dem im folgenden Schema ausgedrückten **Prinzip**:

- **Initialisierung:** Schaffung einer Generation 0
- **Beginn des Zyklus:** Zufällige Auswahl einiger Individuen aus der gesamten Population
- **Schaffen einer neuen Generation** durch die Verwendung folgender Methoden:
  - *Kreuzung:* "Austauschen" von Teilen weniger Individuen
  - *Mutation:* zufällige Veränderung einiger Gene
  - *Reproduktion:* Kopieren von unveränderten Individuen
- **Berechnung der Tauglichkeit** der neuen Generation
- **Beendigung des Zyklus** – Der Vorgang wird von Punkt 2 an wiederholt, bis eine Endbedingung erreicht ist
- **Ende des Algorithmus** – Das Individuum mit der höchsten Leistungsfähigkeit ist die Ausgabe des Hauptalgorithmus und repräsentiert die bestmögliche Lösung

12.2. Terminologie

- **Phänotyp:** Kennzeichnung eines Individuums
- **Erbgut, Genom, Chromosom:** Verkörperung eines Phänotyps
- **Chromosom:** unterteilt in verschiedene, linear angeordnete Gene (*i* -th chromosomische Gene derselben Art, welche dasselbe Merkmal repräsentieren)
- **Allele:** verschiedene Gen-Werte
- **Fitness-Wert:** reicht von 0-1, drückt die Qualität jedes Individuums aus

Jedes Individuum kann auf unterschiedliche Weise kodiert (genetisch beschrieben) werden. Die Beschreibungsmethode entscheidet über Erfolg oder Misserfolg beim Lösen einer speziellen Aufgabe.

12.3. Beispiel:

Generation 0 (Fitness-Wert # "1"):

- 0100011011 *f=0,5*
- 0101000100 *f=0,3*
- 1010110000 *f=0,4*
- 1110111000 *f=0,6*

Auswahl

- *Gewichtete Rollkur-* $p_i = \frac{f_i}{\sum_1^N f_i}$ *ve:*
  - Wahrscheinlichkeit, ein Elternteil zu sein

- *Turnier-Methode*
  - Aus den Gruppen jeder Eltern-Gruppe wird die Person mit dem höchsten Fitness-Wert ausgewählt

- *Beschneidung*
  - Alle Individuen werden gemäß des f-Werts sortiert, der geringwertige Teil wird abgeschnitten, es werden Eltern aus dem restlichen Pool ausgewählt

- *Zufällige Auswahl*
  - Einfachste Methode: der f-Wert spielt keine Rolle in der Auswahl von Individuen für die Aufgabe der weiteren „Erziehung"

- *Kreuzung*
  - Eltern tauschen Teile ihres genetischen Codes aus
  - Die einfachste Methode ist die Einpunkt-Kreuzung
  - Die Stelle für die Beschneidung wird zufällig ausgewählt

  - X: 010001|1011
  - Y: 111011|1000
  - P: 0100011000 *f=0,3*
  - Q: 1110111011 *f=0,8*

  - Die Mehrpunktkreuzung bietet die Möglichkeit, mehr als 2 Eltern zu kreu-

zen

- *Mutation*
  - Zufällige Veränderung zufälliger Erbfaktoren in einem Individuum
  - Sehr geringe Wahrscheinlichkeit

  - 0100011011 ⇒ 0101011011
  - 0101000100 ⇒ 0101100100
  - 1010110000 ⇒ 1010110100
  - 1110111000 ⇒ 1010111000

  - Es ist möglich, Eigenschaften zu erhalten, welche sich nicht in der ursprünglichen Generation finden

Beendigung

- Dieser allgemeine Prozess wird wiederholt, bis die Abschlussbedingungen erreicht sind. Gebräuchliche Abschlussbedingungen sind:

  - Es wird eine Lösung gefunden, welche die Minimal-Kriterien erfüllt
  - Eine festgelegte Anzahl an Generationen wird erreicht
  - Das vorgegebene Budget (Rechenzeit/Geld) wird erreicht
  - Die Fitness der am besten bewerteten Lösung erreicht bzw. erreichte eine Ebene, auf der erfolgreiche Iterationen zu keinen besseren Resultaten mehr führen
  - Manuelle Überprüfung
  - Eine Kombination aus den oben genannte

DATENBANKEN

1. Grundkonzepte von Datenbanken

Eine Datenbank ist eine bestimmte Menge an Informationen (Daten), meistens eine Datensatztabelle, welche auf einem Speichermedium gespeichert ist. Im weiteren Sinne enthält eine Datenbank Software-Ressourcen, welche den Zugriff bzw. die Manipulation der gespeicherten Daten erlauben.

Eine **Datenbank** ist eine Menge von Datensätzen, welche zu einem speziellen Zweck gesammelt wird. Datenbanken werden zumeist für die Speicherung umfangreicher Informationen verwendet. Datenbank-Systeme sind als Teil von Office-Paketen (zB MS Access, OpenOffice.org Basis) verfügbar. Diese Systeme sind auch als eigenständige Programme verfügbar, welche dazu eingesetzt werden, große Radio-Datenbanken zu erstellen. Beispiele hierfür sind MySQL, Oracle und andere.

Eine **Datenbank** ist eine Menge an Daten (Informationen über reale Objekte), welche auf irgendeine Art verbunden sind. Daten sind ein Ausdruck für Daten, welche zur Beschreibung eines Phänomens oder einer Eigenschaft eines beobachteten Objekts verwendet werden. Sie repräsentieren eine Form der Darstellung realer Objekte (Zeichen, Symbole, Bilder, Fakten, Ereignisse), womit sie den Zustand der Realität zu einem bestimmten Zeitpunkt wiederspiegeln.

Eine **Information** ist eine Nachricht darüber, dass ein bestimmtes Phänomen aufgetreten ist. Sie entsteht dadurch, dass Daten eine Bedeutung zugeschrieben wird und steht in einer Beziehung zum Empfänger. Die Nachricht dient dazu, über Veränderungen in der wahrgenommenen Realität zu informieren. Datenbanken begegnen uns im täglichen Leben sehr oft. Daher soll an dieser Stelle aufzeigt werden, wo Datenbanken mit beträchtlichem Ausmaß vorwiegend zum Einsatz kommen:

- Datenbanken für Terminplanungen
- Staatliche Verwaltungsdatenbank
- Informationssysteme von Banken, Schulen und Büros
- Patientenaktensysteme
- Büchereibibliotheken

1.1. Managementsysteme für Datenbanken

Ein Managementsystem für Datenbanken (Abkürzung DBMS in Anlehnung an die englische Bezeichnung database management system) ist eine Softwareausrüstung, welche sicherstellt, dass mit der Datenbank gearbeitet werden kann. Das heißt, sie erstellt eine

Schnittstelle zwischen Anwendungsprogrammen und den gespeicherten Daten. Manchmal wird dieses Konzept mit dem Konzept eines Datenbanksystems verwechselt. Allerdings handelt es sich bei einem Datenbanksystem um einen Verbund aus DBMS und Datenbank.

Entität

Dazu zählt jedes Objekt (Person, Tier, Gegenstand oder Phänomen) der realen Welt, welches im Datenmodell erfasst ist. Eine Entität muss von anderen Entitäten unterscheidbar sein und unabhängig von diesen existieren können.

Daten

Ein Ausdruck für Daten, welche verwendet werden, um eine Phänomen oder eine Eigenschaft eines beobachteten Objekts zu beschreiben. Daten werden durch Messung oder Beobachtung gewonnen und können in zusammenhängende und zugeschriebene Daten unterteilt werden. Zusammenhängende Daten beziehen sich auf eine konstante Skala, bei zugeschriebenen Daten ist dies nicht der Fall.

Information

Informationen sind Daten, welche neue Erkenntnisse bringen.

Datensätze und Attribute

Tabellenzeilen mit Attributwerten für ein Objekt (eine Entität), welche unterschiedlich sein müssen. Datensätze sind Spalten einer Tabelle, Attribute werden ihrem spezifischen Datentyp und Bereich zugeordnet, welcher der Menge zulässiger Werte eines vorgegebenen Attributs entspricht. Die Zeile wird über den Spalten der Tabelle beschnitten und dient dazu, die Daten selbst abzuspeichern.

Attribute, Felder

Eigenschaften, welche bei Entitäten im Blick behalten werden:

- Erstellen von Tabellenspalten
- Diese können verschiedene Werte erhalten
- Die Felder haben einen bestimmten Datentyp (Zahl, Text, Datum, ...)

Primärschlüssel

Dieser identifiziert einmal den Datensatz (Zeile der Tabelle). Es handelt sich um ein Attribut (Feld), welches einen eindeutigen Wert für jede Entität besitzt, zB eine Geburtsnummer, für gewöhnlich ein Hilfsfeld mit einer ID-Nummer.

Fremdschlüssel

Ein Attribut, welches der Primärschlüssel in einer anderen Tabelle ist.

Index

- Dieser stellt eine Möglichkeit dar, eine Tabelle zu sortieren.
- Die Reihenfolge der Datensätze in der Tabelle verändert sich nicht, während die Datenbank „lebt"; der Index hilft dabei, schnell Daten in der Tabelle zu finden.
- Der Index erstellt eine Hilfsdatei mit einer Tabellensortierung anhand eines spezifischen Felds.
- Einige Indizes können, abhängig von verschiedenen Attributen, zu einer Tabelle sortiert werden.
- Der Primärschlüssel ist immer ein Index.

1.2. Datenbankstruktur

Die gebräuchlichsten Datenbanken sind relationale Datenbanken. In diese werden Daten in kleineren Tabellen gespeichert, um eine minimale Redundanz (Datenredundanz) zu gewährleisten. Die Tabellen werden mithilfe von Sitzungen miteinander verbunden. Sitzungen ermitteln die Beziehungen zwischen Tabellen und legen die Zusammenschaltung einzelner Tabellen fest.

Jede dieser Tabellen sollte Daten enthalten, die sich ausschließlich auf einen Objekttyp beziehen (zB Auftragstabelle, Kunden, Preise, Waren etc.). Um eine gute Datenbank zu erstellen, ist es zunächst notwendig, die korrekte Struktur einzelner Tabellen vorzuschlagen. Diese Tabellen müssen dann durch Sitzungen verknüpft werden. Tabellen bilden die Grundlage der gesamten Datenbankstruktur. Die Grundregeln für die Tabellengestaltung lauten wie folgt:

- Jede Information sollte nur einmal in der Datenbank vorkommen
- Jede Tabelle sollte Informationen über einen Objekttyp enthalten
- Wenn Tabellen erstellt werden, sollte das Ausmaß künftig noch zu ergänzender Daten berücksichtigt werden

1.3. Datenbanktabelle

Eine Tabelle sollte Informationen über einen Objekttyp enthalten. Die Datenbanktabelle ist einer normalen Tabelle recht ähnlich. Zeilen enthalten Datensätze (über ein Objekt) und Spalten geben Datenwörter oder Felder an. Das Feld wird manchmal als eine Überschneidung von einer Zeile und einer Spalte bezeichnet, welches eine einen einzelnen Wert (ein Datenelement) enthält.

Zaměstnanci

ID_zamestn	Jméno	Příjmení	Datum naro	Pohlaví	Telefonní čí	ID_funkce
1	Tomáš	Novák	28.4.1980	muž	723 123 456	F_03
2	Josef	Koblížek	15.3.1976	muž	728 452 123	F_01
3	Petra	Maková	5.2.1985	žena	724 556 115	F_02
4	Václav	Sýkorka	30.6.1970	muž		F_06
5	Denisa	Rosolová	12.12.1956	žena		F_05
6	Michal	Aspik	3.6.1976	muž		F_04
7	Dominik	Kokeš	14.2.1981	muž		F_04
8	Tereza	Železná	25.10.1978	žena		F_02
9	Vladimíra	Mimořádná	17.11.1989	žena		F_02
10	Jakub	Pekelník	5.12.1973	muž		F_05

In der oben abgebildeten Tabelle erstellen MitarbeiterInnen Zeilen mit Datensätze, welche Informationen über individuelle MitarbeiterInnen enthalten. Die Spalten repräsentieren die Felder, wo man stets einen Datentyp (Text, Datum) sieht. Jede Spalte (jedes Datenwort) hat einen Namen, einen ausgewählten Datentyp (zB Text, Zahl, Ja/Nein, Datum und Zeit) und eine bestimmte Größe. Weitere Merkmale (Format, Standard, etc.) können ihr zugeschrieben werden. Mehr Informationen über diese Angelegenheit erfährt man in den Videoleitfäden.

1.4. Primärschlüssel

In den meisten Fällen ist es notwendig, jeden Eintrag in der Tabelle eindeutig zu identifizieren. Zu diesem Zweck wird der sogenannte Primärschlüssel herangezogen. Ein Primärschlüssel ist ein Datenbereich (Array), welcher die Absicht verfolgt, die eindeutige Indentifikation individueller Datensätze in einer Tabelle sicherzustellen. Der Primärschlüssel ist für gewöhnlich ein Einzelfeld (ein sogenannter einfacher Primärschlüssel), aber er kann auch ein aus mehreren Tabellen bestehendes Array (ein sogenannter zusammengesetzter Primärschlüssel) sein.

Es kann in jeder Tabelle nur einen Primärschlüssel geben. Mithilfe des Primärschlüssels wird schneller nach Informationen gesucht. Darüber hinaus werden Beziehungen zu anderen Tabellen hergestellt. Die Werte im Primärschlüsselfeld müssen in jedem Datensatz eindeutig sein. Der Primärschlüssel ist einer der Indizes. Für eine schnellere Rück-

gewinnung und Sortierung von Datensätzen entsprechend eines bestimmten Tabellenfelds, ist es ratsam, sich der Feldindexierung (Indizes) zu bedienen. Wenn nach einer anderen Tabellenspalte als dem Primärschlüssel gesucht wird, wird diese zum Index bestimmt. Durch das Festlegen eines Index' wird die Sortierung, Suche und Bearbeitung von Werten in den Tabellen beschleunigt.

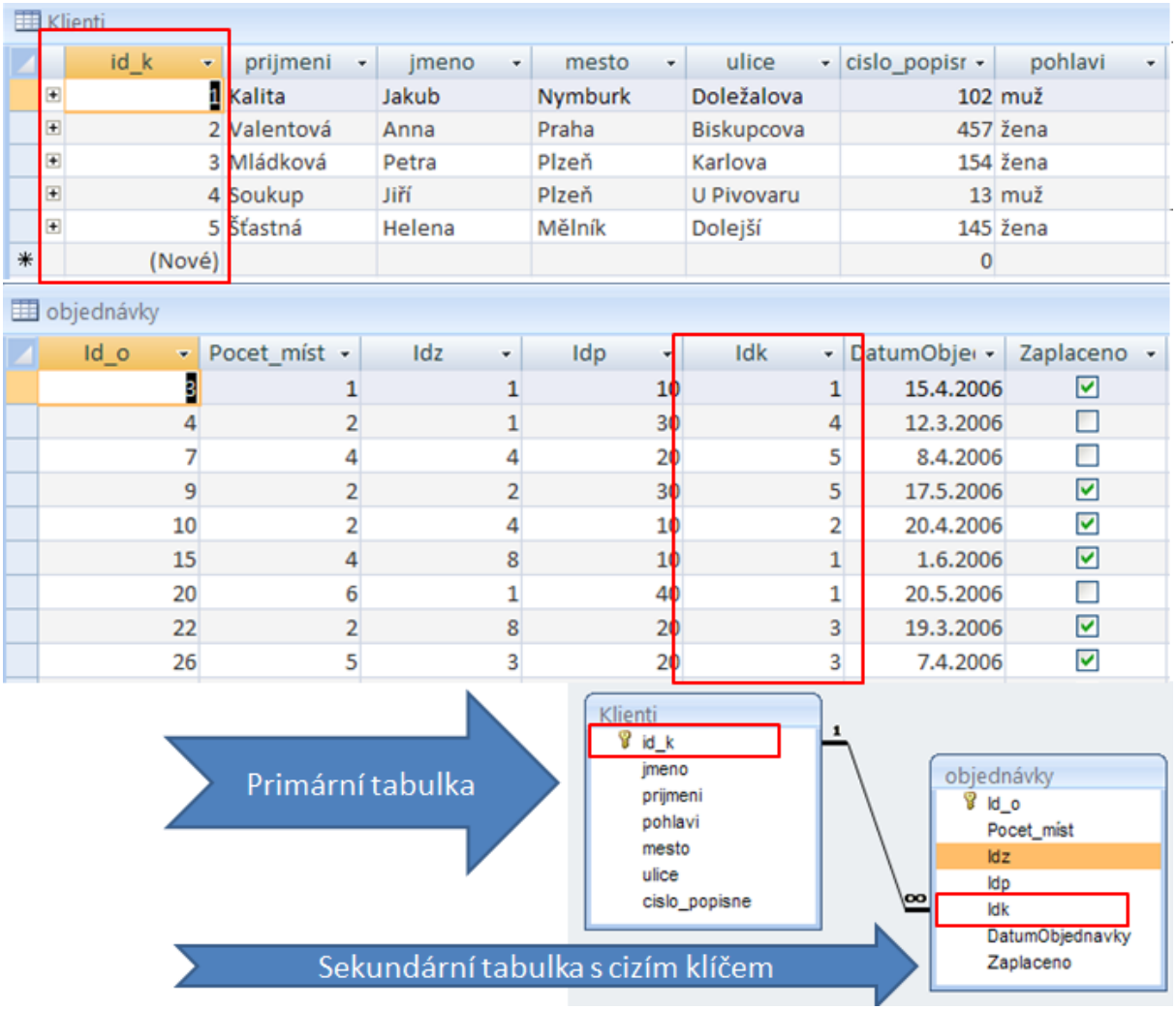
1.5. Beziehung

In einer relationalen Datenbank sind einzelne Tabellen via Sitzungen verknüpft. Der Hauptzweck der Beziehung zwischen Tabellen ist es, das Auftreten redundanter Daten einzuschränken. Daten sollten nicht wiederholt in verschiedenen Tabellen einer einzelnen Datenbank eingespeist werden. Die Beziehung basiert auf der Verbindung zwischen dem einzigartigen Feld (Primärschlüssel) einer Tabelle und dem sich darauf beziehenden Feld einer anderen Tabelle.

Um eine Beziehung zwischen zwei Tabellen herzustellen, bedarf es eines speziellen Fels in jeder Tabelle. In einer dieser Tabellen ist der Primärschlüssel zu finden. Diese Tabelle bezeichnet man gemeinhin als Primärtabelle. In der zweiten Tabelle wird ein spezielles Feld zu Sitzungszweck erstellt. Dieses Feld wird als Fremdschlüssel bezeichnet und enthält die Werte des Primärschlüssels einer anderen Tabelle. Eine Tabelle, welche einen Fremdschlüssel enthält, wird als Sekundärtabelle bezeichnet.

1.6. Primär- und Sekundärtabelle

Das Primärschlüsselfeld enthält ausschließlich eindeutige Werte. Das Fremdschlüsselfeld kann dieselben Werte enthalten. Das Primär- und das Fremdschlüsselfeld müssen dieselben Werte enthalten, um eine korrekte Sitzung zu erstellen. Die nachfolgende Grafik zeigt eine Primärtabelle, welche einzelne Kundendatensätze enthält. Der Primärschlüssel ist das erste Feld, welches die Kundennummer enthält. Diese Tabelle ist mit der Sekundärtabelle verknüpft, welche aus Datensätzen mit Kundenbestellungen besteht. Der Fremdschlüssel in der Sekundärtabelle ist das Feld, welches die Anzahl der Kunden enthält, die bereits eine Bestellung aufgegeben haben. Aus dem oben angeführten Beispiel lässt sich herauslesen, dass ein Datensatz in der Primärtabelle mit einem oder mehreren Datensätzen der Sekundärtabelle übereinstimmt. Anders gesagt: ein Kunde kann eine oder mehrere Bestellungen aufgeben. Dies entspricht dem Sitzungstyp 1: N.



1.7. Arten von Sitzungen

Es werden drei Grundarten von Sitzungen unterschieden:

Beziehungsart 1:1 (außergewöhnlich): Ein Datensatz der Primärtabelle stimmt nur mit einem Datensatz der Sekundärtabelle überein. Ein Beispiel wäre, dass eine Person nur einer Geburtsnummer zugeordnet ist.

Beziehungsart 1:N (gebräuchlichste): Ein Datensatz der Primärtabelle stimmt mit einem Datensatz oder mehreren Datensätzen in der Sekundärtabelle überein. Diese Art wurde bereits im oben angeführten Beispiel mit den Kunden und Bestellungen erklärt.

Beziehungsart M:N (sehr oft): Ein Datensatz oder mehrere Datensätze in der Primärtabelle stimmen mit einem Datensatz oder mehreren Datensätzen in der Sekundärtabelle

überein. Diese Sitzungen werden mithilfe einer Kopplungstabelle hergestellt, welche über zwei Sitzungen des Typs 1: N mit den Originaltabellen verknüpft wird.

1.8. Verweisintegrität

Die Verweisintegrität hält die Beziehungen zwischen den Tabellen aufrecht. Sie erlaubt es nicht, dass Datensätze in die Sekundärtabelle eingefügt werden, für die es keinen übereinstimmenden Datensatz in der Primärtabelle gibt. Sie überwacht auch die Veränderung von Fremdschlüsselwerten, wenn der Primärschlüssel verändert wird. Innerhalb der Verweisintegrität ist es möglich, Regeln für das Löschen von Datensätzen festzulegen.

1.9. Funktion der Datenbank

Professionelle Datenbanken enthalten eine große Menge wichtiger Informationen. Personen, die mit so einer Datenbank arbeiten, können in verschiedene Gruppen unterteilt werden:

- ein Datenbankspezialist plant und erstellt professionelle Datenbanken
- ein Benutzer fügt Daten ein, wartet die Daten und ruft Informationen aus der Datenbank ab
- der Datenbank-Manager gewährt Benutzern Zugang zu bestimmten Daten und ist für die Rettung der Datenbank nach einem Zusammenbruch oder schlimmen Fehler verantwortlich.

2. Datenbankmodelle

Ausgehend von der Sichtweise, wie Daten und Verknüpfungen zwischen diesen gespei-chert wer-den, wird bei Datenbanken zwischen zwei Grundarten unterschieden:

2.1. Hierarchische Datenmodelle

Daten sind in Form einer Baumstruktur organisiert. Jeder Datensatz entspricht einem Knoten in dieser Baumstruktur, die Beziehung zwischen den Datensätzen ist vom Typ Eltern/Nachkomme. Das Aufspüren von Daten in hierarchischen Datenbanken erfordert es, dass man durch Datensätze hin zum Kindknoten, zurück zum Elternknoten oder auf die Seite hin zum nächsten Sprossknoten navigiert. Der größte Nachteil hierarchischer Layouts ist, dass das Einfügen und Löschen von Datensätzen komplexe Operationen erfordert. Auch die zeitweise unnatürliche Organisation der Daten ist zuweilen ein Nach-teil.

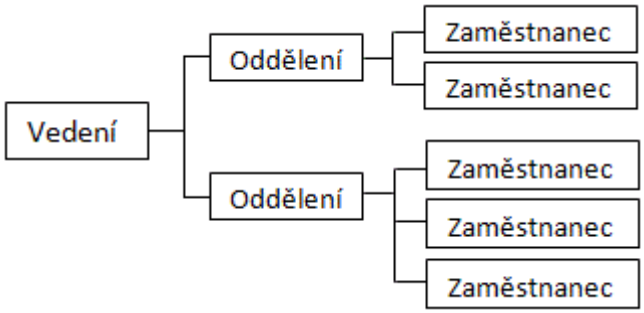


Abbildung: Hierarchisches Modell

2.2. Netzwerkdatenmodelle

Das Netzwerkdatenmodell ist im Wesentlichen eine Verallgemeinerung des hierarchi-schen Modells, welches dieses um Mehrfachbeziehungen (Sets) ergänzt. Diese Sets ver-binden Datensätze desselben oder eines anderen Datentyps miteinander, wobei die Verbindung zu einem oder mehreren Datensätzen hergestellt wird. Der Zugriff auf ver-knüpfte Datensätze erfolgt unmittelbar ohne vorhergehende Suche; es gibt dafür Ope-rationen: Ausfindig machen des Schlüsseldatensatzes, verschieben des ersten Nach-wuchsknotens in das Sub-Set, verschieben des anderen Sprossknotens in das Set, Auf-rücken vom Nachwuchs- zum Elternknoten in einem anderen Set. Die Nachteile einer Netzwerk-Datenbank sind deren besondere Starrheit und die Schwierigkeit, deren Struk-tur zu verändern.

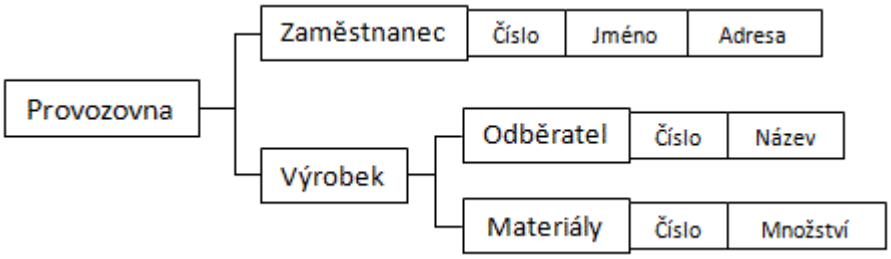


Abbildung: Netzwerkmodell

2.3. Rationales Datenmodell

Das relationale Datenbankmodell gehört zu den jüngsten und meistverwendeten Modellen. Zurzeit wird es am häufigsten vom Ministerium für Wirtschaft und Industrie. Das Model hat eine einfache Struktur, Daten sind in Tabellen organisiert, die aus Zeilen und Spalten bestehen. Alle dieser Datenbankoperationen werden in den Tabellen ausgeführt.

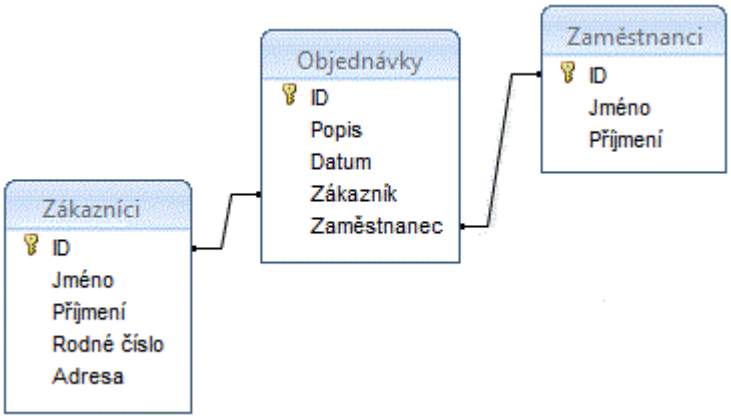


Abbildung: Relationales Modell

Eine relationale Datenbank muss die folgenden zwei Charakteristika besitzen:

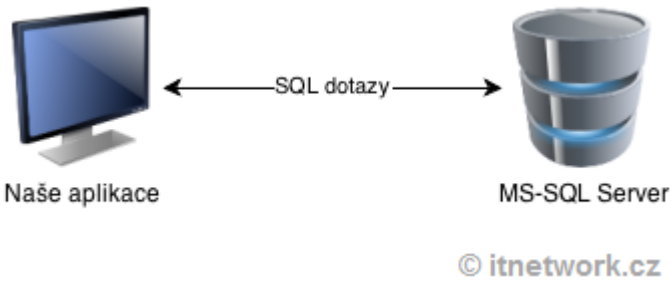
- die Datenbank wird vom Benutzer als eine Menge von Sitzungen und nur als diese wahrgenommen
- die Auswahl-, Prognose- und Verbindungsoperationen sind im relationalen ROI verfügbar, ohne einen explizit vordefinierten Zugangspfad zu benötigen, um diese Operationen auszuführen.

2.4. Objektdatenbanken

Neben relationalen Datenbanken gibt es auch noch die bereits erwähnten Objektdatenbanken. Diese befassen sich mit der Problematik der Objekt- und beziehungsbezogenen Inkompatibilität. Sie stellen denselben Komfort wie ORMs zur Verfügung, müssen jedoch intern keine Daten in Tabellen konvertieren, da diese als Objekte abgespeichert werden. In der Theorie gibt es keinen Leistungs-oder einen anderen Grund, warum sie nicht über kurz oder lang relationale Datenbanken ersetzen sollen. In der Praxis jedoch werden sie so gut wie kaum verwendet und man muss hoffen, dass sich das mit der Zeit ändert.

2.5. Angehängte Anwendung

Man kann die Zugriffsanwendung verwenden, wenn man Daten in Echtzeit lesen oder ändern möchte. Durch die Verwendung von DataReader, Command und Connection-Klassen werden SQL-Statements direkt an SQL geschickt, was unmittelbar zum Erhalt von Ergebnissen führt.
Diese Situation ist auf der nachfolgenden Grafik abgebildet:



2.6. Objektdatenbanken

Managementsysteme für Datenbanken (DBMS) sind derzeit die am weitesten verbreiteten Werkzeuge zur Speicherung und Manipulation von Daten kommerzieller Anwendungen. Mit ihnen lassen sich große Datenbanken relativ einfach managen, darüber hinaus bieten sie ein hohes Maß an Integrität, eine effiziente Suche und Verarbeitung von Informationen. Sie tolerieren Fehler und Ausfälle, ermöglichen einen gleichzeitigen Zugriff mehrerer Benutzer und bieten noch weitere Vorteile.

Das relationale Datenmodell – das gebräuchlichste Modell in den heutigen Datenbanksystemen – schränkt die Struktur und Beziehungen der gespeicherten Daten auf ein kleines Set von Tabellen ein, welches über einem vordefinierten Set an einfachen Datentypen liegt. Der unbestrittene Vorteil dieses Modells ist dessen Einfachheit und – daraus resultierend – die einfache Standardisierung und Portierbarkeit. Der Nachteil hingegen

ist, dass sich das reale Datenschema vom internen Datenbank-Tabellenmodell unterscheidet. Alle Beziehungen zwischen den Daten können nur in Form von Tabellen ausgedrückt werden, was in einer Anwendung mit einem etwas komplexeren Datenmodell zu einer Anzahl an Tabellen führt, die über Hilfsverknüpfungen, sogenannten Keys, miteinander verbunden sind.

Dies resultiert im Verlust von Klarheit, die Datenbank lässt sich schlechter managen und zukünftige Veränderungen im Datenmodell der Anwendung zwingen Programmierer dazu, merklich in ihre Tabellen-Darstellung einzugreifen. Relationale Datenbanken sind und werden auch weiterhin das bevorzugte Mittel sein, um kommerzielle Anwendungen zu managen, die dadurch gekennzeichnet sind, dass sie eine über eine große Menge an Daten mit einfacher Struktur verfügen. Viele Anwendungen wie z.B. Design-, Multimedia-, geografische System usw. brauchen hingegen ein Datenmodell, welches dafür sorgt, dass komplexe reale Daten besser mit deren Darstellung im Datenbanksystem korrespondieren. Besagtes Modell ist das Objektdatenmodell, welches in Datenbankensystemen auf bekannten Grundsätzen des objektorientierten Modellierens und Programmierens basiert. Darüber hinaus ist es um Techniken der Ausdauer, Darstellung von Beziehungen, Befragungen, Transaktionszugriffen usw. angereichert.

2.7. Grundlagen der Objektorientierung – Objekte und Klassen

Ein objektorientiertes Modell basiert auf dem Grundsatz, Informationen aus der realen Welt in Objekte zu zerlegen. Ein Objekt bezeichnet gemeinhin jede (sogar strukturierte) Entität, die in einem bestimmten Kontext der sie umgebenden Welt einzigartig und unabhängig identifizierbar ist. Daher hat das Objekt eine eindeutige Identität und alle zwei oder weiteren ähnlichen, baugleichen Objekte grenzen sich klar voneinander ab. Die Identität eines Objekts wird durch einen Objektidentifizierer (OID) festgestellt, welcher von dem System erstellt wird, einzigartig ist, sich während der Existenz eines Objekts nicht verändert und sowohl für den Programmierer als auch den Endverwender versteckt ist. Objekte werden durch Klassen charakterisiert. Die Klasse ist eine zusammenfassende Beschreibung des Objekts, sie ermittelt die Datenordner des Objekts und die Operationen (auch Methoden genannt), die in diesem Objekt durchgeführt werden können. Jedes Objekt ist die Instanz einer Klasse. Es ist grundsätzlich möglich, eine uneingeschränkte Anzahl von Instanzen strukturell identischer Objekte in einer Klasse zu erstellen. Als nächstes wird die Schnittstelle der Klasse gekennzeichnet.

2.8. Literale

Neben Objekten wird auch das Konzept der Literale in einem objektorientierten Modell vorgestellt. Ein Literal ist eine Datenentität eines besonderen Datentyps, welcher – an-

ders als ein Objekt – keine eigene Identität besitzt. Literale scheinen für gewöhnlich als Objektattribute auf. Die Menge der Operationen, welche im Datentyp eines Literals durchführbar sind, ist festgelegt, sie kann nicht verändert werden. Objekte und Literale stehen in engem Zusammenhang mit dem Konzept der Variabilität (Mutabilität). Variablität wird als die Fähigkeit verstanden, Daten zu verändern und gleichzeitig deren Identität zu erhalten. In diesem Sinne sind Objekte variabel, da es möglich ist, die Werte ihrer Datenkomponenten zu verändern, während sie gleichzeitig ihre ursprüngliche Identität beibehalten. Literale hingegen sind nicht variabel.

2.9. Operation

Es gibt einige Grundarten von Operationen in Objekten Jedes Objekt hat eine oder mehrere Konstruktoren. Der Zweck eines Konstruktors ist es, ein Objekt im Zuge ihrer Erstellung mit einem Feld vorzubelegen. Darüber hinaus ist auch ein Destruktor Teil jedes Objekts. Der Destruktor wird abgerufen, wenn das Objekt unterbrochen wird. Sein Zweck ist es, das Objekt zu reinigen, bevor es entfernt wird. Eine wichtige Operation in Objekten ist das Kopieren. Hierbei werden die sogenannten seichten und tiefen Kopien unterschieden. Im Gegensatz zur seichten Kopie werden bei der tiefen Kopie nicht nur die Attribute eines Objekts kopiert, es werden auch Kopien von Objekten erstellt, auf welche das ursprüngliche Objekt verwies. Andere typische Operationen sind Methoden zum Herausfinden und Zuordnen von Attributwerten, Methoden zur Durchführung von Berechnungen und Manipulation von Objektattributen, Methoden um eine Nutzerausgabe zu produzieren usw.

Ein **Objektdatenmodell** entspricht den in einer Objektstruktur gespeicherten Daten. Es handelt sich für gewöhnlich um die Trennschicht eines Objekts zwischen dem Code und der Datenbank, wo die Daten, mit welchen die Anwendung arbeitet, nicht den Datenbankserver mit Anfragen belasten.

Objektbezogenes Datenmodell

Das objektbezogene Datenmodell ist eine klassische Datenbank, welche durch abstrakte Datentypen (ADT) erweitert wird. ADTs sind nutzerdefinierte Typen, welche aus rudimentären Datentypen einer Datenbank bestehen. Was verletzt 1NF?

- Objektmerkmale werden nun in alle größeren SARBs implementiert.
- Objekttpyen und deren Methoden werden zusammen mit Daten in der Datenbank gespeichert, damit der Programmierer keine ähnlichen Strukturen in jeder Anwendung erstellen muss.
- Der Programmierer kann auf ein Set von Objekten zugreifen, als ob es sich dabei um ein Objekt handeln würde.
- Objekte können einfach Verbindungen bilden, welchen zufolge eine Entität aus anderen Entitäten besteht (ohne, dass man dafür Verbindungen bräuchte).

- Methoden laufen auf dem Server. Es gibt keine ineffiziente Datenübermittlung über das Netzwerk.

Objektdatentypen

können folgendes enthalten:

- Daten – Attribute
- Operationen – Methoden
- ORSRBD-spezifisch ist der Umstand, dass man sich Daten sowohl aus relationaler Sicht (Datensätze, Operationen) als auch aus Objektsicht anschauen kann.

Arten von Methoden:

- Mitgliedermethoden – sie werden in einem bestimmten Objekt aufgerufen.
- Statische Methoden – sie werden in dem Datentyp aufgerufen.
- Konstruktor – ein Standard – Konstruktor wird für jeden Objekttyp festgelegt.

3. Integrität von Datenbanken

Die Integrität einer Datenbank bedeutet, dass sich die darin gespeicherten Daten konsequent an vorgegebene Regeln halten. Nur Daten, welche den vordefinierten Kriterien entsprechen, können eingegeben werden (so müssen z.B. der für die Tabellenspalte vergebene Datentyp berücksichtigt oder andere zulässige Einschränkungen berücksichtigt werden). Integrierte Einschränkungen dienen dazu, diese Integrität sicherzustellen. Es handelt sich hierbei um Werkzeuge, welche verhindern, dass falsche Daten eingefügt werden oder bestehende Datensätze verloren gehen oder beschädigt werden, wenn man mit der Datenbank arbeitet. So ist es mitunter möglich sicherzustellen, dass Daten gelöscht werden, welche bereits ihren Sinn verloren haben. Ein Beispiel hierfür wäre es, dass beim Löschen eines Benutzers auch die damit verknüpften Datensatzrestbestände in anderen Datenbanktabellen entfernt werden.

3.1. Arten von Integritätseinschränkungen

Einschränkungen der Entitätsintegrität: Verpflichtende Integritätseinschränkung um sicherzustellen, dass der Primärschlüssel einer Tabelle vollständig ist (verhindert die Speicherung von Daten, welche mit Daten in einer anderen Zeile der Tabelle identisch wären).

Einschränkungen der Domainintegrität: Diese stellen sicher, dass die Datentypen/Datendomains eingehalten werden, welche für die Spalten der Datenbanktabellen definiert wurden.

Einschränkungen der referenziellen Integrität: Diese beschäftigen sich mit den Beziehungen zwischen zwei Tabellen, in welchen ihre Zusammenhänge durch die Verbindung zwischen Primär- und Fremdschlüssel festgelegt wird.

Aktive Hinweisintegrität: Diese definiert die Aktivitäten, welche eine Datenbank auszuführen hat, wenn Regeln verletzt werden.

3.2. Berücksichtigung von Integritätseinschränkungen

Grundsätzlich gibt es drei Wege sicherzustellen, dass Integritätseinschränkungen eingehalten werden

1. Die Stelle auf dem Server der Datenbank, wo einfache Mechanismen zur Wartung von Integritätseinschränkungen gespeichert werden

- Dies ist der beste Weg, Daten zu schützen
- Es dauert allerdings länger, bis der Benutzer eine Antwort vom System erhält und kann nicht immer für ein anderes Datenbanksystem sichergestellt werden

2. Die clientseitige Speicherstelle für Schutzmechanismen

- Ist aufgrund Komforts und der Unabhängigkeit des Datenbanksystems die beste Wahl
- Der Bedarf an Kontrollmechanismen für jede Operation kann Anwendungsfehler verursachen, weshalb mehrere Anwendungen repariert werden müssen.

3. Separate serverseitige Programmmodule

- In modernen Datenbanksystemen werden zu diesem Zweck so genannte Trigger implementiert. Diese entsprechen separaten Verfahren, die automatisch vor und nach Operationen laufen können, welche Daten behandeln
- Dies ermöglicht die Implementation komplexer Integritätseinschränkungen
- Der Nachteil ist hier wieder die stark eingeschränkte Möglichkeit der Migration auf ein anderes Datenbanksystem auf dem Server

Die ideale Lösung ist die Kombination der vorher genannten Möglichkeiten in Abhängigkeit von den besonderen Bedingungen. Die Überprüfung von Integritätseinschränkungen werden für gewöhnlich nach jeder Operation durchgeführt, was die Anforderungen an den Server reduziert. Es ist nicht notwendig zu protokollieren, welche Überprüfungen später durchgeführt werden müssen. Allerdings können komplexere Integritätseinschränkungen nicht immer verifiziert werden, weshalb sich die Konformität erst nach dem Abschluss der gesamten Transaktion überprüfen lässt.

3.3. Beziehungen zwischen Tabellen

Sitzungen werden verwendet, um Daten zu verbinden, die miteinander in Beziehung stehen und sich in verschiedenen Datenbanktabellen befinden. Grundsätzlich unterscheidet man zwischen vier Arten von Beziehungen:

- Es gibt keinen Zusammenhang zwischen Tabellen, weshalb keine Beziehungen definiert werden

- 1:1 (ein Datensatz stimmt nur mit einem Datensatz in einer anderen Datenbanktabelle über-ein und umgekehrt)

- 1:N (ordnet einen Datensatz mehreren Datensätzen einer anderen Tabelle zu)

- Dies ist die gebräuchlichste Art einer Sitzung, da sie sich mit vielen Situationen im echten Leben deckt

- M: N (erlaubt es, mehrere Datensätze einer Tabelle mehreren Datensätzen einer zweiten Tabelle zuzuordnen)

- Aus praktischen Gründen wird diese Beziehung zumeist realisiert, indem zwei Beziehungen der Art 1: N und 1: M kombiniert werden, welche auf eine unterstützende, so genannte Kopplungstabelle referenzieren, welche aus einer Kombination beider verwendeter Schlüssel besteht.

3.4. Normale Formen

Der Begriff Normalisierung bezeichnet das Verfahren der Vereinfachung und Optimierung der vorgeschlagenen Strukturen von Datenbanktabellen. Das Hauptziel ist es, Datenbanktabellen so zu gestalten, dass sie ein Minimum an redundanten Daten enthalten. Die Korrektheit der Struktur kann mithilfe der folgenden normalen Formen evaluiert werden.

Nullte Normale Form (0NF)

- Die Tabelle enthält mindestens eine Spalte (ein Attribut) welches mehrere Arten von Werten enthalten kann.

Erste Normale Form (1NF)

- Alle Tabellenspalten können nicht mehr weiter in Teile aufgeteilt werden, die Informationen enthalten – Elemente gleichen Atomen
- Eine Spalte enthält keine zusammengesetzten Werte.

Zweite Normale Form (2NF)

- Die Tabelle enthält nur Spalten, welche vom gesamten Schlüssel abhängen

Dritte Normale Form (3NF)

- Die Tabelle befindet sich in der dritten normalen Form, wenn es keine Abhängigkeiten zwischen Spalten gibt

Vierte Normale Form (4NF)

- Die Spalten in der Tabelle beschreiben nur einen Tatbestand oder einen Kontext

Fünfte Normale Form (5NF)

- Durch die Ergänzung einer neuen Spalte würde sich die Tabelle in mehrere Tabellen aufteilen

3.5. Datenbankintegrität

Datenbankintegrität bedeutet, dass sich die Datenbank an spezifizierte Regeln hält, konkret an Integritätseinschränkungen. Diese Integritätseinschränkungen sind Teil der Datenbankdefinition und das Managementsystem für Datenbanken ist verantwortlich für deren Einhaltung.

Feste Einschränkungen können sich entweder auf einzelne Werte beziehen, welche in die Felder einer Datenbank eingegeben werden (zB eine Note für ein Lehrfach muss zwischen 1 und 5 liegen) oder eine Bedingung für die Kombination von Werten in einigen Feldern eines Datensatzes darstellen (zB darf das Geburtsdatum nicht später sein als das Todesdatum). Eine ganzzahlige Einschränkung kann auch für eine ganze Menge an Datensätzen einer bestimmten Art gelten. So kann es zB sein, dass ein bestimmtes Feld oder eine Kombination von Feldern innerhalb der gesamten Datensatzmenge in einer Datenbank, welche einer bestimmen Art sind, einen eindeutigen Wert haben muss (ein Beispiel hierfür wäre die ID-Nummer in einem Personendatensatz).

Häufig verwendete Integritätseinschränkungen in relationalen Datenbanken beziehen sich auf die referenzielle Integrität. Es handelt sich hierbei um eine Anforderung, der zufolge ein Datensatzfeld einen Verweis auf einen anderen Datensatz irgendwo in der Datenbank enthalten muss und der referenzierte Datensatz tatsächlich existiert, sodass diese Verknüpfung nicht ins Leere führt und keine sogenannte Datenbankwaise darstellt.

Feste Einschränkungen

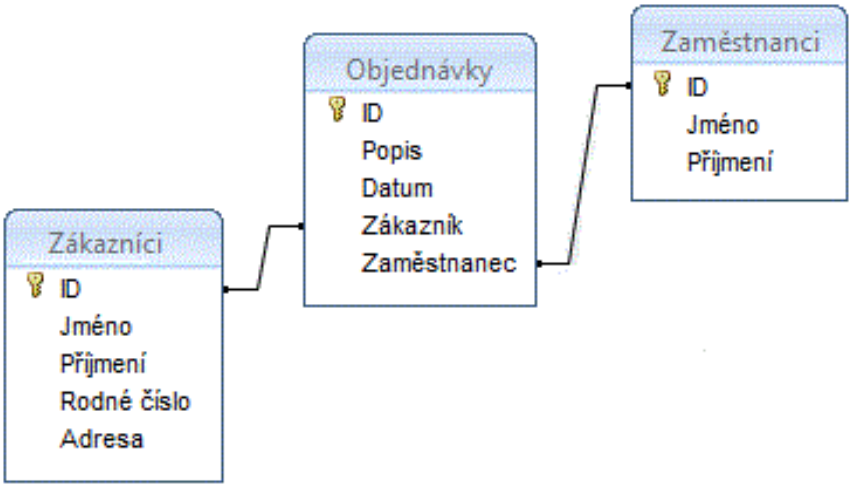
Es sollte sichergestellt werden, dass nur Datensätze, die Beziehungen der realen Welt entsprechen, in die Datenbank aufgenommen werden (so darf ein Altersattribut keine negativen Zahlen erhalten). Um mit solchen Fällen richtig umzugehen, wird eine Integritätseinschränkung verwendet, mit deren Hilfe eine Bandbreite von Werten festgelegt wird, die ein bestimmtes Attribut annehmen kann. Feste Einschränkungen spezifizieren, welche Einschränkungen und interne Beziehungen Datenbank-Daten einhalten müssen. Sitzungen, die den Integritätseinschränkungen entsprechen, sind erlaubt.

Relationales Datenbanksystem

- muss ein effizientes Management und effiziente Analyse gespeicherter Daten ermöglichen
- muss die folgenden drei Grundfunktionen erfüllen:
    - das Definieren von Datentypen erlauben
    - die Arbeit mit Daten ermöglichen
    - das System sollte die Mittel enthalten, um erforderliche Arbeitsgänge mit Daten auszuführen, zB Daten sortieren, filtern, Berechnungen mit Daten anstellen, Datenmanagement
- im Besonderen überprüft das System den Zugang zu den Daten, sprich wer autorisiert ist, auf die Daten zuzugreifen und wer sie aktualisieren darf
    - das System steuert die Datenteilung unter mehreren Benutzern

4. Relationales Datenbankmodell

Abbildung: Relationales Modell



Eine relationale Datenbank muss die folgenden zwei Merkmale erfüllen:

- Die Datenbank wird vom Benutzer ausschließlich als eine Menge von Sitzungen wahrgenommen
- Zumindest Operationen zur Auswahl, Hochrechnung und Verknüpfung sind im relationalen ROI verfügbar, ohne dass sie explizit vordefinierte Zugangspfade benötigen, um diese Operationen auszuführen.

4.1. Zwölf Regeln gelten für das relationale Datenbankmodell

1. Informationsregel:

Jede Information in einer relationalen Datenbank wird explizit auf logischer Ebene und nur eine Weise ausgedrückt: in Form von Werten in der Tabelle.

2. Sicherheitsregel:

Auf alle Daten in der relationalen Datenbank kann man garantiert zugreifen, wenn man die Tabellennamen mit den Werten des Primärschlüssels und dem Namen der Spalte kombiniert.

3. Systematische Verarbeitung von Null-Werten:

Null-Werte werden von einem Managementsystem für Datenbanken vollständig unterstützt, um undefinierte Informationen abzubilden, unabhängig vom Datentyp.

4. Dynamischer on-line Katalog basierend auf dem relationalen Modell:

Die Beschreibung der Datenbank erfolgt auf logischer Ebene auf dieselbe Weise wie jene von Kundendaten, weshalb ein autorisierter Benutzer dieselbe relationale Sprache für seine Anfrage verwenden kann, wenn er mit den Daten arbeitet.

5. Regel zur Konfiguration von Interpretationen:

Alle Interpretationen, die theoretisch möglich sind, können auch im System konfiguriert werden.

6. Ein umfassendes Daten Sub-Wörterbuch:

Ein relationales System kann mehrere Sprachen und verschiedene Modi für Begriffsoperationen unterstützen. Allerdings muss es zumindest eine Befehlssprache mit einer ausgeklügelten Syntax geben, welche allgemeine Datendefinitionen, Interpretationsdefinitionen, Interaktivität und Programmmanipulationen, Integritätseinschränkungen, autorisierten Datenbankzugriff, Trans-aktionsbefehle usw. zulässt.

7. Fähigkeit zum Einfügen, Erstellen und Löschen von Daten:

Die Fähigkeit, relationale Regeln von sowohl standardmäßigen als auch abgeleiteten Sitzungen zu warten, bleibt nicht nur bei der Sichtung von Daten aufrecht, sondern auch bei Operationen zum Penetrieren, Ergänzen und Löschen.

8. Unabhängigkeit physischer Daten:

Anwendungsprogramme sind unabhängig von der physischen Datenstruktur.

9. Unabhängigkeit logischer Daten:

Anwendungsprogramme sind unabhängig von Veränderungen in der logischen Struktur einer Datenbankdatei.

10. Integrale Unabhängigkeit:

Feste Einschränkungen müssen von relationalen Datenbankressourcen oder deren Sprache definiert werden. Es muss möglich sein, diese im Katalog anstatt im Anwendungsprogramm zu speichern.

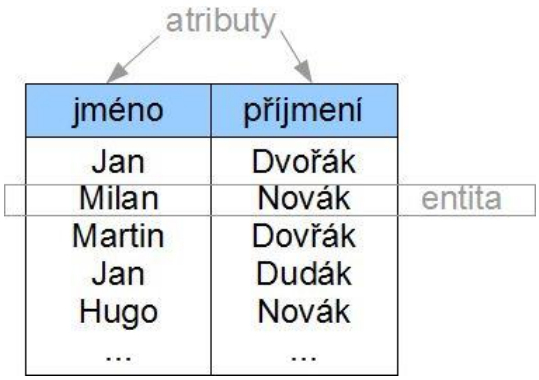
11. Unabhängigkeit der Distribution:

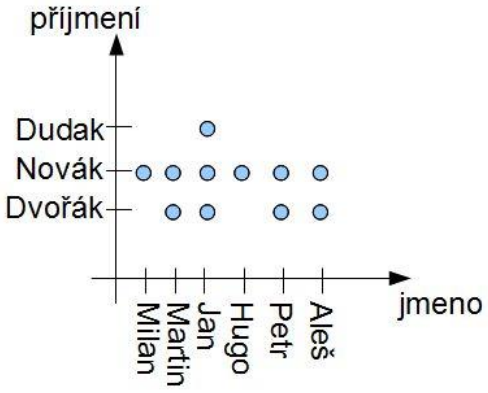
Relationale RBDs müssen in der Lage sein, auf anderen Rechnerarchitekturen aufzusetzen.

12. Datenbankzugriffsregel:

Wenn das relationale System ein niedriges Sprachniveau hat, kann dieses Niveau nicht verwendet werden, um Integritätseinschränkungen zu erstellen. In diesem Fall muss sich das System in Form einer höherstufigen relationalen Sprache ausdrücken.

4.2. Relationales Datenmodell





Ein relationales Datenmodell ist eine Möglichkeit, Daten in Tabellen zu behalten. Der Begriff relational wurde gewählt, weil die Tabelle in Form einer Sitzung definiert ist.

Die Beziehung ist im Wesentlichen eine Tabelle. Sie ist als Untermenge eines kartesianischen Produkts einer Domain definiert. Die Beziehung auf der rechten Grafik ist daher eine Untermenge des Sets von {Dudák, Novák, Dvořák} x {Milan, Martin, Jan, ..., Aleš}

Anders als bei einer mathematischen Sitzung verändert sich die Datenbank mit der Zeit (durch das Hinzufügen und Entfernen von Sitzungselementen). Neben standardmäßigen Set-Funktionen stößt man hier auf eine Auswahl-Funktion zur Zeilenauswahl und Projektion sowie zur Spaltenauswahl bei Datenbanksitzungen.

Eine **Domain** ist die Menge all jener Werte, die ein Attribut annehmen kann. Anders gesagt, die Bandbreite von Attributwerten. In der Praxis wird die Integrität einer Domain eingeschränkt (IO).

Die **Bildannahmeattributsdomain** ist die Menge aus {Dudak, Novák, Dvořák}. * Note

Ein **Attribut** ist die Eigenschaft einer Entität. Aus Sicht einer Tabelle handelt es sich um eine Spalte.

Das **relationale Schema** kann als Struktur einer Tabelle (Attribute und Domains) verstanden werden.

Beispiel:

Tabelle (Sitzung): ein Lehrer

Attribute: ID, Vorname, Zuname, Funktion, Büro

Domains:

- D1 – Drei Buchstaben vom Familiennamen, Anzeige von drei Zeichen. Zahlen.
- D2 – Bezeichnung des Kalenders
- D3 – Set von Familiennamen
- D4 – Set von Funktionen (Assistent, Wissenschaftler, Lehrer ...)
- D5 - A101, A102, ... A160

Relationales Schema: Lehrer (ID, Name, Adresse, Funktion, Büro)

Sitzung: Lehrer = {(nov001, lukas, novak, scientist, A135), (com123, jan, comenius, teacher, A111)

Sie ist definiert als R (A, f), wobei A einer Menge von Attributen entspricht (A1, A2, ... An) und die function f (Ai) = Di das Domainattribut zuordnet.

4.3. Relative Datenmodelleigenschaften

Nachfolgend werden die Tabelleneigenschaften einer Sitzungsdefinition aufgelistet:

- Spaltenhomogenität (Domainelemente)
- Jeder Wert (der Wert eines Attributs in der Spalte) ist ein atomarer Eintrag
- Die Reihenfolge der Zeilen und Spalten spielt keine Rolle (es handelt sich um festgelegte Elemente/Attribute)
- Jede Zeile einer Tabelle ist eindeutig identifizierbar durch den Wert eines oder mehrerer At-tribute (Primärschlüssel)

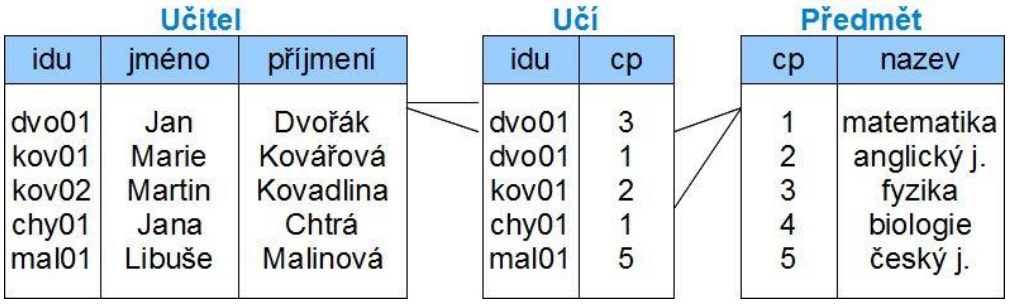
4.4. Beziehungen im Relationalen Modell

Gemeinhin werden Verbindungen im relationalen Modell mithilfe anderer Sitzungen umgesetzt. Hierbei handelt es sich um eine sogenannte Kopplungstabelle. Diese enthält jene Sitzungsattribute (inklusive der Verbindungen), welche eindeutig deren Entitäten identifizieren – die Primärschlüssel. Wenn eine Tabelle ein Attribut enthält, welches in einer anderen Tabelle als Primärschlüssel dient, so enthält sie einen Fremdschlüssel. Die Kopplungstabelle enthält deshalb Fremdschlüssel.

In der Grafik unten sieht man die Sitzungen „Lehrer“ mit Primärem ID-Schlüssel und die Sitzung mit Primärem Schlüssel cp. Damit wird die Beziehung Lehrer UČÍ ausgedrückt. Das Lehrfach wurde als die neue Sitzung „Lernen“ erstellt, welche zwei Fremdschlüssel

enthält (idu bezieht sich auf den Lehrer und cp bezieht sich auf die Entität des Lehrfachs). Nun ist es möglich, den Lehrer über die Kopplungs-tabelle mit jenem Lehrfach zu verbinden, das er unterrichtet.

Warum dem Lehrer kein Lehrfachattribut in der Tabelle zugewiesen wurde? Ganz einfach deshalb, weil ein Lehrer mehr als ein Lehrfach unterrichten kann. Zwischen dem Lehrer und dem Lehrfach besteht eine Beziehung M: N. Selbst wenn man „Lehrer" der Lehrfach-Attributstabelle hinzufügt, würde das nicht der Beziehung entsprechen (ein Lehrfach kann von mehreren Lehrern unterrichtet werden). Bei einer 1: N-Beziehung ist das möglich, und diese kommt auch in der Praxis vor. Die Kopplungstabelle ist nur notwendig, um die M: N-Beziehungen umzusetzen.



5. SQL Grundlagen

5.1. Einführung in SQL

SQL – Structured Query Language – ist ein allgemeines Instrument zur Manipulation, Verwaltung und Organisation von Daten, welche in einer Computerdatenbank gespeichert sind. Sie ist vor allem für Benutzer gedacht, werden aber auch auf verschiedene Arten von Anwendungsentwicklern verwendet. Sie kann an jede Umgebung angepasst werden.

Der Name dieses Werkzeugs ist kurz, aber nicht unkompliziert. SQL ist nicht nur eine Abfragesprache, sie kann auch Daten definieren, sprich eine Tabellenstruktur, Spalten einer Datentabelle (Array) mit Daten fühlen und Beziehungen zwischen Datenobjekten sowie deren Organisation definieren. Sie ermöglicht darüber hinaus eine Datenzugriffskontrolle, sprich das Gewähren und Entziehen von Zugriffsprivilegien auf verschiedenen Ebenen. Dadurch können Daten vor versehentlicher oder intendierter Zerstörung, unautorisierter Betrachtung oder Bearbeitung sowie deren Weitergabe an Dritte geschützt werden. Überdies wird eine reibungslose Funktionalität sichergestellt, wenn mehrere Benutzer gleichzeitig auf die Daten zugreifen.

SQL ist eine spezielle Programmiersprache, welche in einer geeigneten Umgebung zum

Einsatz kommt – und zwar entweder benutzerfreundlich oder interaktiv zum sofortigen Erledigen von Aufgaben (meistens Anfragen). Zuweilen werden ihre Befehle auch in die Wirts-Programmsprache eingefügt. Allerdings handelt es sich bei SQL nicht um eine vollwertige Programmiersprache, zum Beispiel deshalb, weil es in den meisten Anwendungen keine Steuerungsprogrammkonstrukte gibt. Darüber hinaus fehlt es noch an anderen Elementen, die jede allgemeine Programmiersprache enthalten sollte. SQL ist daher ein standardisiertes Werkzeug, um mit relationalen Datenbanken zu arbeiten. Sie ist kein Datenbanksystem, aber ein anderweitig integrierter Teil eines Verwaltungssystems für Datenbanken.

SQL ist vor allem eine interaktive Abfragesprache: Sie ermöglicht es, quasi in Echtzeit Antworten auf sehr komplizierte Abfragen zu bekommen. Sie ist kein prozesstechnisches Werkzeug mit einer Menge an Datenzugriffen und ist eine standardisierte Sprache. SQL ist nachvollziehbar, da es Daten in Tabellenform versteht, weshalb sie für Benutzer leicht zu verstehen ist. SQL arbeitet mit relationalen Datenbanken, in welchen dem Benutzer Daten als eine Menge verschachtelter Tabellen angezeigt werden. Jede Tabelle stellt eine Datenmenge dar, welche in Form von Zeilen (Einträgen) und Spalten (Objekte) angeordnet ist. Der Benutzer nimmt einen Datenwert als Element in einer Matrix wahr.

In der überwiegenden Mehrheit der Fälle ist das Resultat einer in SQL beschriebenen Aufgabe eine Datenmenge aus einer Tabelle oder mehreren Tabellen, eine sogenannte Ergebnistabelle, welche nicht immer das letztendliche Produkt sein muss. Sie kann als eine Menge von Eingabedaten für die weitere Verarbeitung dienen, zB zum Drucken eines Etiketts oder zum Zeichnen eines Diagramms.

SQL kann als „gängige Sprache“ dienen, vor allem wenn sie in Netzwerken operiert, in welchen verschiedenen Datenbankprodukten verwendet werden.

SQL wird auch verwendet, um Vorschauen (Abfragen) zu erstellen. SQL Vorschauen ermöglichen es, verschiedene Ansichten von Tabellenstrukturen und Daten für verschiedene Benutzer zu erstellen. Jeder Benutzer sieht nur die Daten, die für ihn bestimmt sind. Die Daten werden von dem Benutzer wieder als einfache Tabelle gesehen, obwohl die Daten in Wahrheit von einer anderen Tabelle stammen. Die angezeigten Daten in der Ansicht sind dynamisch. Werden die Daten in den Tabellen (Datenbankdateien) hin- und herbewegt, ändern sich auch die Daten, welche die Vorschau anzeigen. Umgekehrt ist es dasselbe. Arbeitet man mit einer Aktualisierungsvorschau (eine spezielle Art der Vorschau, die nicht nur die Anzeige von Daten, sondern auch deren Ergänzung und Aktualisierung erlaubt) und verändert man dort die Daten, werden die Veränderungen in einer geeigneten Tabelle (Datenbankdatei) wiedergegeben.

Eine Schlüssel-SQL-Aussage ist ein Befehl. Jeder Befehl beginnt mit einem Schlüsselwort. Das Wort gibt an, zu welcher Aktivität der Befehl führt. Auf das Schlüsselwort folgen ein oder mehrere optionale Klauseln, welche die Natur der zu erledigenden Tätigkeit oder

die Daten, mit deren Hilfe ein Befehl ausgeführt werden soll, spezifizieren.

Jeder Abschnitt beginnt mit einem Schlüsselwort, wie zB FROM oder WHERE. Einige Klauseln sind verpflichtend, andere optional. Jede SQL-Anwendung verwendet – neben den standardmäßig von den ANSI/ISO-Konventionen vorgegebenen Klauseln – ihre eigenen Klauseln. Manchmal unterscheidet sich die Funktionsweise von Standardklauseln geringfügig.

SQL-Objektnamen sind in der Regel zwischen einem und 18 Zeichen lang, das erste Zeichen muss ein Buchstabe sein und die Bezeichnung darf keine Leerzeichen oder Interpunktion enthalten. Wenn man ein Objekt mit einem Namen bezeichnet, ist es oft notwendig, diesen Namen einzuschränken. Dies ist dann der Fall, wenn es mehrere gleiche Namen gibt und nicht klar ist, auf welches Objekt sich ein Name bezieht. Eine Namenseinschränkung wird mittels `.´(Punkt)-Vermerk vorgenommen. Sehr häufig werden Einschränkungen bei Tabellenbezeichnungen eingesetzt:

```
<Owner>. <Table_name>
```

Oder wenn Spalten eingeschränkt werden, was in Datenbanksystemen gang und gäbe ist.

`<Tablename>. <Columnname>`, in der Regel sind Einschränkungen auch in SQL erlaubt.

Auf mehreren Ebenen:

```
<Owner>. <Table_name>. <Column_name>
```

5.2. SQL Abfragen

Eine Einführung in Aussagen in SQL-Datenbanken.

Einsatz von SQL-Befehlen

Zunächst einmal gilt es, standardmäßige SQL-Statements in einfachen Beispielen einzusetzen. Die so erzeugten Informationen können dann in der Programmierung von Webseiten, in Visual Basic oder Delphi verwendet werden. Die allgemeine Einführung in Datenbanken ist im Einleitungsartikel zu finden. Wenn man die Datenbankentheorie bereits versteht, kann man sich im nächsten Schritt anschauen, wie eine Datenbank in Microsoft Access erstellt wird.

Liste mit SQL-Statements

Die wichtigsten Befehle sind: SELECT, INSERT, DELETE, CREATE, FROM, WHERE und viele andere, welche in den folgenden Kapiteln vorgestellt werden. Zunächst wird jedoch eine einfache Tabelle erstellt, in welcher besagte Befehle eingesetzt werden (um klar zu machen, wozu jeder Befehl dient).

Erstellen einer Übungstabelle

zamestanci : Tabulka

ID	id_zam	jmeno	prijmeni	funkce	kancelar	plat	vek
1	1	Pavel	Velky	vyvoj	vyvojova	10000	20
2	2	Roman	Maly	vyvoj	konstrukcni	12000	25
3	3	Petr	Vezir	konstrukter	konstrukcni	15000	35
4	4	Adela	Nevečeřelova	učetní	finanční	10000	20
5	5	Martin	Šéf	učetní	finanční	18000	23
6	6	Tomáš	Bílý	ředitel	ředitelství	100000	35
7	7	Miloslava	Bílá	asistent	ředitelství	50000	16
8	8	Ferdinand	Velky	topič	ředitelství	25000	66
9	9	Kamila	Novaková	nástěnkář	ředitelství	35000	25
10	10	Laura	Benešova	sekretař	konstrukcni	25000	55
11	11	Jarmil	Pražsky	finance	zlodejovna	30000	36
12	12	Jarmila	Jungová	ekonom	zlodejovna	35000	28
13	13	Dalibor	Navratil	ekonom	zlodejovna	32000	19
14	14	Norbert	Mikulec	skladnik	sklad	2000	48
15	15	Iveta	Vezirova	konstrukter	vyvojova	18000	47
(Nové číslo)	0					0	0

Záznam: 1 z 15

Um die Begriffe im SQL-Skript zu erklären empfiehlt es sich, diese in ein paar einfachen Tabellen zu testen. Sobald man sie einmal verstanden hat, lassen siche sich in einer 10.000 Zeilen umfassenden Tabelle einsetzen. Diese Tabelle kann mittels MySQL, FoxPro, Acces oder einem anderen Datenbankprogramm erstellt werden.

SELECT Aussage

Ein wichtiger Befehl. Wir verwenden unsere erstellte Tabelle und probieren die Anwendung dieses Befehls aus. Die einfachste Form der Aussage sieht wie folgt aus:

```
SELECT *
FROM employees;
```

Listet die Inhalte der gesamten Tabelle „Employees“ auf.

```
SELECT the name
```

```
FROM employees;
```

Listet die Inhalte der Spalte „Name“ aus der „Employees“-Tabelle auf.

```
SELECT DISTINCT function
FROM employees;
```

Listet die Inhalte der Spalte „Funktion“ in der „Employees“-Tabelle auf, zeigt aber nur die Objekte ohne Duplikate an (auch nur dann einmal, wenn der Entwickler 4x gelistet hat).

```
SELECT AVG(plat)
FROM zamestanci;

SELECT SUM (flat)
FROM employees;
```

Die Spalte AVG berechnet die Steigung einer vorgegebenen Spalte, die Spalte SUM berechnet die Summe der Werte einer vorgegebenen Spalte, oder es können auch in der Abfrage selbst Berechnungen durchgeführt werden.

```
SELECT SUM (PLAT + 500)
FROM employees;
WHERE
```

Wenn man 10.000 Zeilen hat, wird man die Berechnungen auf ein paar Kriterien beschränken wollen. Dies wird mithilfe von WHERE zusammen mit ein paar mathematischen Symbolen erreicht.

```
SELECT name
FROM employees
WHERE pay> 20000;
```

Listet die Namen der Mitarbeiter auf, die mehr als 20.000 Tschechische Kronen (ca. 800 Euro) verdienen.

Gebräuchliche einfache Abfragesyntax

Wählt (SELECT) die Liste von Spalten aus der WHERE-Tabelle der WHERE-Suchbegriff um nach den Sortierspalten zu ordnen (ORDER).

Die Spaltenliste ist entweder eine durch Kommas geteilte Liste von Spalten oder ein * Zeichen, um alle Spalten auszuwählen. Der Spaltenname ist entweder nur der Spaltenname oder der columname.class_name. Es kann auch ein Tabelleneintrag * verwendet werden, um alle Spalten aus der Tabelle auszuwählen. Die Tabelle kann jede der folgenden sein:

- Sitzung (echte Datenbanktabelle)
- Ergebnis einer anderen SELECT-Abfrage
- Ansicht
- Ergebnis des JOIN-Operators (virtuelle Tabelle)

Der Suchbegriff ist eine Bedingung, die jeder Datensatz erfüllen muss, welcher in den Abfrage-Ergebnissen auftaucht. Selbstverständlich muss ein Datensatz erst in der Quelltabelle existieren, bevor dieser geschrieben werden kann. Deshalb ist WHERE eine einschränkende Bedingung, es sei denn sie wird aufgelistet, sodass alle Zeilen in der Tabelle geschrieben werden. Die Column-Spalten sind eine Liste durch Kommas abgegrenzter Spalten, gemäß welcher die Ergebnistabelle sortiert wird. In der ersten Tabelle ist das primäre Sortierungskriterium, die zweite Spalte ist das sekundäre Sortierungskriterium, wenn man nicht auf Basis der ersten Spalte entscheiden kann usw.

Beispiel 1

Eingabe: Wählen Sie alle Personen mit dem Namen Radka und Tomas aus der Datenbank aus.

Lösung 1

```
SELECT name, surname, nickname FROM person
WHERE name = 'Radek' ORname = 'Tomas'
```

Man kann gebräuchliche logische Operatoren AND und OR verwenden, in den meisten Datenbanken geht stattdessen auch die Schreibweise && und ||. Es lässt sich das gebräuchliche = verwenden, um Zeichenketten (Strings) zu vergleichen, wobei es ein paar Dinge zu bedenken gilt. Abhängig von den Servereinstellungen kann die Zeichenlänge von Bedeutung sein oder nicht. Die Standard-Einstellung (und deshalb häufiger) ist, dass die Zeichenlänge egal ist. Der Akela.mendelu.cz-Server unterscheidet Groß- und Klein-

schreibung. Es hängt auch von den Einstellungen des Textvergleichsserver (und der Datenbank) ab, ob Akzente beachtet werden oder nicht. Akela.mendelu.cz berücksichtigt ferner auch diakritische Zeichen, weshalb der Name exakt so geschrieben werden muss, wie er in der Datenbank gespeichert ist.

Lösung 2

SELECT name, surname, nickname FROM persons WHERE name IN ('Radek', 'Tomas')

Eine Alternative zur vorherigen Abfrage ist die Verwendung des IN-Operators. IN ist ein festgelegter Operator, welcher überprüft, ob sich ein Wert in einer spezifizierten Menge befindet. Das heißt, dass aus jeder Zeile der Wert der Namensspalte genommen und daraufhin getestet wird, ob sich dieser in der vordefinierten Menge befindet (‚Radek', ‚Tomas'). Die Menge kann entweder mittels direkter Enumerierung von Werten oder durch eine andere SQL-Abfrage definiert werden.

Beispiel 2

Eingabe: Wählen Sie alle Menschen in der Datenbank aus, die in Prag leben.

Lösung – Schritt 1

```
SELECT id_address, FROM FROM WHERE address mesto = 'Praha'
```

Im ersten Schritt werden alle Adressen eruiert, die sich in Prag befinden. Zu diesem Zweck empfiehlt es sich, das Datenbank-Schema abzurufen, welches offenbart, dass es eine Spalte id_address in der Personentabelle gibt, welche mit der Spalte id_address in der Adress-Tabelle verknüpft ist. Der Grund dafür ist folgender: Werden alle Personen ausgewählt, deren Adresse im Ergebnis der oberen Abfrage aufscheint, erhält man genau jene Personen, deren Adresse sich in Prag befindet.

Lösung – Schritt 2

```
SELECT name, receive, nickname FROM people
WHERE id_addresses IN ()

SELECT DISTINCT id_address FROM WHERE address
mesto = 'Praha' ()
```

Nun gibt es zwei SELECT-Konstruktionen in der Abfrage. Das zweite SELECT ist das die sogenannte Sub-Abfrage und es handelt sich um eine leichte Modifikation der Abfrage, welche im ersten Schritt erstellt wurde. Die erste Anpassung ist, dass sie nun nur die Spalte id_address auswählt. Dies ist angesichts der Tatsache, dass die Eltern-SQL-Abfrage nach ID_Adressen sucht, essentiell. Die Sub-Abfrage darf daher nur die ID_Adressen zurückgeben. Zusätzlich arbeitet der IN-Operator nur mit einer Skalar-Menge, weshalb eine Sub-Abfrage (zur Vorgabe einer Menge für den IN-Operator) stets nur eine Spalte zurückgeben darf. Werden mehrere Spalten in der Sub-Abfrage verwendet, wird der Datenbankserver folgende Fehlermeldung erhalten: ERROR: Sub-Abfrage hat zu viele Spalten. Die zweite Veränderung ist, dass das Schlüsselwort DISTINCT hinzugefügt wird. Dadurch wird sicher-gestellt, dass wie zurückgegebenen Werte eindeutig sind (siehe die nächsten Beispiele). In diesem besonderen Fall ist es unnötig (da die id_address immer einmalig ist). Trotzdem ist es ratsam, DISTINCT einzubauen. Einerseits handelt es sich dabei um eine allgemeine Vorbeugungsmaßnahme, andererseits tendiert der IN dazu bei Sub-Abfragen, in welchen sich die Werte wiederholen (dann ist es keine Menge), sich seltsam zu verhalten.

6. SQL – Komplexere Abfagen

Übungen: SQL-Abfragen mit mehreren Tabellen

6.1. Beispiel 1

Eingabe: Schreiben Sie eine SQL-Abfrage, welche aus der Datenbank alle Personen auswählt, die eine ICQ-Nummer haben. Wählen Sie den Vornamen, den Familiennamen und die ICQ-Nummer der Person aus. Sortieren Sie in aufsteigender Reihenfolge nach dem Familiennamen.

Möglichkeit 1

Die Leichteste: berechnet, dass der Wert von id_type_contact = 1 mit dem ICQ-Typ in der contact_type-Tabelle übereinstimmt. Der JOIN-Operator arbeitet stets mit zwei Kalkulationstabellen. Die Kopplungsbedingung ist immer, dass sich Werte in ein paar Tabellen gleichen. Die join-Bedingung muss beide Tabellen enthalten, die sich im JOIN-Abschnitt befinden. Die join-Bedingung muss Spalten enthalten, welche die Tabellen verbinden. Es gibt keinen Verweis auf die Kontakte-Tabelle in der Personen-Tabelle. Die Kontakte-Tabelle ist nur ein einzelner Verweis auf die Personen-Tabelle – die Spalte id_object. INNER JOIN wählt aus der Tabelle nur jene Datensätze aus, die der Verbindungsbedingung entsprechen. Es handelt sich dabei ausschließlich um jene, welche einen Kontakt haben (aber da der Suchbegriff in der Abfrage spezifiziert wurde, ist es in diesem speziellen Fall auch möglich, LEFT und RIGHT JOIN zu verwenden, wobei es sich hierbei nur um einen überein-stimmenden Umstand handelt).

```
SELECT person.name, person.name, contact.contact FROM
     INNER JOIN contacts
          ON contacts.id_osoby = person.id_osoby
WHERE contacts.id_types_contact = '1'
ORDER BY by ASC
```

Möglichkeit 2

Identisch zur vorherigen Möglichkeit, wobei sie USING verwendet. Wenn die Verknüpfungsbedingung die Gleichwertigkeit gleichgenannter Spalten (id_id) ist, kann deren Eingabe vereinfacht werden.

```
SELECT person.name, person.name, contact.contact FROM
     Persons JOIN contacts USING (id_persons)
WHERE contacts.id_types_contact = '1'
ORDER BY by ASC
```

Möglichkeit 3

Die bessere Möglichkeit: Wählt Kontakte auf Basis des Kontaktartnamens aus, weshalb sie nicht auf irgendeinen id_type_contact Wert angewiesen ist. Der Kontaktartnamen befindet sich in der contact_type-Tabelle. Es ist wichtig zu bedenken, dass das Ergebnis verbundener Tabellen eine Tabelle ist und dass der JOIN-Operator immer mit zwei Tabellen arbeitet.

```
SELECT person.name, person.name, contact.contact FROM
     (INNER JOIN contacts
          ON contacts.id_osoby = person.id_osoby)
               INNER JOIN contact_types
                    ON contact_types.id_type_contact =
                    contacts.id_types_contact
WHERE contact_name.nazev = 'icq'
ORDER BY by ASC
```

SELECT ist darüber hinaus ein Befehl, der viele andere Worte haben kann, einige Tabellen ineinander verschränken kann usw. Aber die gesamte Grundlage der Syntax ist: SELECT, dann die Liste der Spalten, welche man erhalten oder kumulieren möchte, oder andere Funktionen FROM und der Name der Tabelle, aus welcher man Daten extrahieren möchte, WHERE und der Ausdruck mit Beschränkungen, welche für die gewünschten Zeilen gelten sollen. In diesem Fall soll die Stadt-Spalte genau den Wert „Chomutov" enthalten.

Zusammen mit der Kumulierungsfunktion sieht dies wie folgt aus: Man will herausfinden, wie viele Zeilen das sind, welche Personen aus Chomutov entsprechen:

```
SELECT COUNT (*) FROM lide WHERE mesto = ”Chomutov”;
```

Es wird eine Zeile in einer Spalte wiedergegeben, welche die Zahl 2 enthält.

Die Bedingungen können mit den boolean‘schen Operatoren AND und OR, Klammern und SQL-Funktionen viel komplizierter sein. Eine Liste dieser ist in der Dokumentation zu finden. Welche Personen aus Chomutov über 30 Jahre alt sind, findet man auf folgende Weise heraus:

```
SELECT * FROM lide WHERE mesto = ”Chomutov” AND age> 30;
```

Ein Sternchen bedeutet „alle Spalten“.

Andere Tabellen werden mithilfe von JOIN mit einer SELECT-Abfrage verknüpft. Dieses Unterfangen erfordert mehr als eine Tabelle und eine Form der Beziehung zwischen ihnen. So braucht man zB die bereits vorliegende Menschentabelle, wo jede Person eine eindeutige ID hat, sowie eine Gefahren-Tabelle mit einer Spalte clovek_id und einer Herausforderung. In den Spalten der Personen wären Zahlen die Personen, welche auf die Idee kamen, und in der Spalte der Idee wäre der Text einer Idee. Anschließend könnte man eine Abfrage erstellen, welche alle Ideen auflisten und jeder davon eine Spalte namens „Mensch“ hinzufügen würde. Aber Delaily geht über diese Vorstellung hinaus:

```
SELECT napady.napad, lide.jmeno FROM napady
LEFT JOIN lide ON lide.id = napady.clovek_id;
```

Das Ergebnis wird eine Tabelle mit Ideen sein, wo es sich bei der ersten Spalte um die Idee und bei der zweiten um die Person handeln wird, von der diese kam.

Wenn man „DELETE“ statt „SELECT (Spalten)“ eingibt, erhält man eine Abfrage zur Löschung von Zeilen. Sie funktioniert gleich wie SELECT (hier gibt es eine WHERE Bedingung), aber statt Ergebnisse zu erhalten, werden die Zeilen, welche mit der Bedingung übereinstimmen, gelöscht. Um Adam zu löschen, muss man nur folgendes eingeben:

```
DELETE FROM lide WHERE name = ”Adam”;
```

Eine der am häufigsten verwenden SQL-Abfragen ist immer noch UPDATE. Diese wird eine Zeile bearbeiten. Wenn Catherine ein Jahr älter ist, wird dies wie folgt aktualisiert:

```
UPDATE people SET age = 30 WHERE name = ”Catherine”;
```

Es können auch mathematische Ausdrücke und Verweise auf bestehende Spaltenwerte verwendet werden. In diesem Fall lässt sich die Spalte Alter wie folgt um die Zahl eins erhöhen:

```
UPDATE people SET age = age + 1 WHERE name = ”Catherine”;
```

Das sind soweit die am häufigsten verwendeten SQL-Abfragen (Klauseln). Individuelle, komplexere und präzisere Abfragen können mehr Wörter enthalten. Um diese zu verstehen ist es jedoch notwendig, sich intensiver mit den individuellen Unterlagen zu beschäftigen oder nach einem spezifischen Problem zu suchen. Verwendet man z.B. die Kumulierungsfunktion in SELECT, erhält man das Ergebnis der Kumulierung der gesamten Tabelle. Möchte man hingegen die Durschnitts-Temperatur der letzten Jahre ermittelt und hat dafür die Monatstemperaturen zur Verfügung, ist es notwendig, GROUP BY zu verwenden. Wenn man bei Google „GROUP BY YEAR DATETIME MYSQL“ eingibt, findet man heraus, dass man DATETIME YEAR benötigt:

```
GROUP BY YEAR (record_date)
```

Neben GROUP BY kann man auch ORDER BY auswählen, wo man spezifiziert, nach welchen Spalten die Ergebnisse sortiert werden sollen und ob dies ab- oder aufsteigend geschehen soll. Manchmal ist die Satzstellung wichtig. Für SELECT findet man diese in den Unterlagen und man sieht überdies, dass man zuerst GROUP BY und dann ORDER BY verwenden muss.

7. GRAPH-DATENBANKEN

Ein Graph ist eine Datenstruktur, die aus Ecken und Kanten besteht. Die Graph-Datenbank ist ein Datenspeicher und -verarbeitungssystem in Form eines Graphen. In vielen Fällen werden Domänen sehr natürlich modelliert: So enthalten solche Domains menschliche Beziehungen, Gene und Proteine, mobile Netzwerke, Distributionsnetzwerke verschiedener Art, neuronale Netzwerke oder ökologische Netzwerke, welche die Interaktionen zwischen Organismen erfassen.

Graphische Datenbanken enthalten oft verschiedene Systeme, welche Daten in Graphen verwalten. Ausgehend von dieser Definition wäre es theoretisch möglich, sich relationale Datenbanken anzusehen, bzw. deren Überbau wie bei Graphen-Datenbanken. Relationale Datenbanken hingegen erlauben keine effektive Speicherung und Abfrage von Graphendaten.

7.1. Rationale Datenbanken

Der ansteigende Pfad in einer relationalen Datenbank erfordert JOINS und ist deshalb ineffizient für tiefe Arbeitsgänge. Ein JOIN, welcher den Index verwendet, erfordert eine logarithmische Zeit O (log n), ohne dass der Index O ist (n log n).
|
So können zB menschliche Beziehungen mithilfe von zwei Tabellen modelliert werden: Man und Beziehung. Für jeden Durchlauf durch die Beziehung zwischen zwei Personen ist in der Regel ein JOIN notwendig. Traditionelle relationale Datenbanken sind für JOIN-Einheiten optimiert. Da es etliche JOINs gibt, bekommen relationale Datenbanken trotz der Verwendung von Indizes Probleme.

7.2. True Chart Datenbanken

Als echte Graph-Datenbanken werden zum Zweck dieses Artikels jene Datenbanken bezeichnet, die nur eine konstante Zeit O (1) für den Durchlauf des Graphen. True Chart Datenbanken speichern Daten, wie andere NoSQL-Datenbanken auch, in einer denormalisierten (bereits „fusionierten“) Form. Für echte Graph-Datenbanken sind sie deshalb besonders wichtig für das Schreiben, während der Sendeabruf minderwertiger ist.

Ein Beispiel für eine True Chart Datenbank ist Neo4j, welche seit mehr als einem Jahr entwickelt wird und weit genug ausgereift ist, um Produkte zu entwickeln. Ein weiteres Beispiel ist das neuere OrientDB System. Beispiele für Datenbanken, welche Graphendaten speichern und abrufen können, aber nicht notwendigerweise effizient Diagramme

durchstöbern, sind die meisten Dreiergruppen (Sesame, Jena, Virtuoso) oder FlockDB (Twitter Projekt).

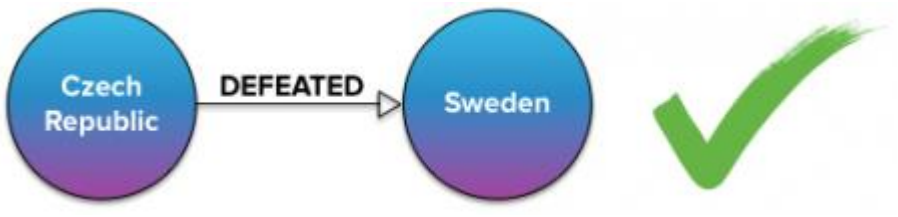
Wie bei den meisten NoSQL-Datenbanken gibt es hier keine einfache Abfragesprache für Graphen-Datenbanken. Allerdings implementieren viele Chart Datenbanken die Gremlin Chart Crawling Sprache, welche in Zusammenarbeit mit den Neo4j-Autoren entwickelt wurde.

Der Übergang von der relationalen Welt in die Welt der Graphen erfordert eine Veränderung im Denken sowie bei der Betrachtung von Daten. Obwohl Graphen viel häufiger intuitiver sind als Kalkulationstabellen, unterlaufen Menschen beim Modellieren solcher Diagramme ständig Fehler. In diesem Artikel werden die häufigsten Fehler thematisiert, ebenso wie die Modellierung von wechselseitigen Beziehungen. Zum Schluss wird auch ein echtes Beispiel gezeigt, welches dazu dient, die Reihe fortzusetzen.

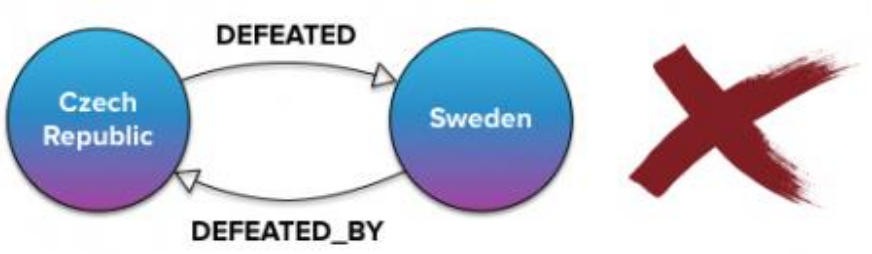
7.3. Einseitige Beziehungen

Beziehungen in Neo4j müssen von einer Art sein, welche der Beziehung eine semantische Bedeutung gibt und die auch eine festgelegte Richtung hat. Dadurch wird der Sinn unter Entitäten ausgedrückt. Anders gesagt ist die Beziehung mehrdeutig, wenn man nicht die Richtung der Beziehung festlegt.

So zeigt das folgende Diagramm zB an, dass die Tschechische Republik Schweden bei einem Eishockey-Spiel besiegt (DEFEATED) hat. Wäre die Richtung der Beziehung umgekehrt, wäre Schweden viel glücklicher. Es gibt keine Möglichkeit zu erfahren, wer der Gewinner ist, weshalb die Beziehung ergebnislos ist.



Man bemerke, dass die Beziehung in gegensätzlicher Richtung existiert, wie es unten im nächsten Graphen gezeigt wird. Hierbei handelt es sich um einen typischen Fall. Noch einmal wird dies anhand eines anderen Beispiels erklärt: Pulp Fiction wurde von Quentin Tarantino inszeniert (DIRECTED), was gleichbedeutend damit ist, dass Quentin Tarantino der Regisseur des Films Pulp Fiction ist (IS_DIRECTOR_OF). Dies kann in zahlreichen Paarungen resultieren.

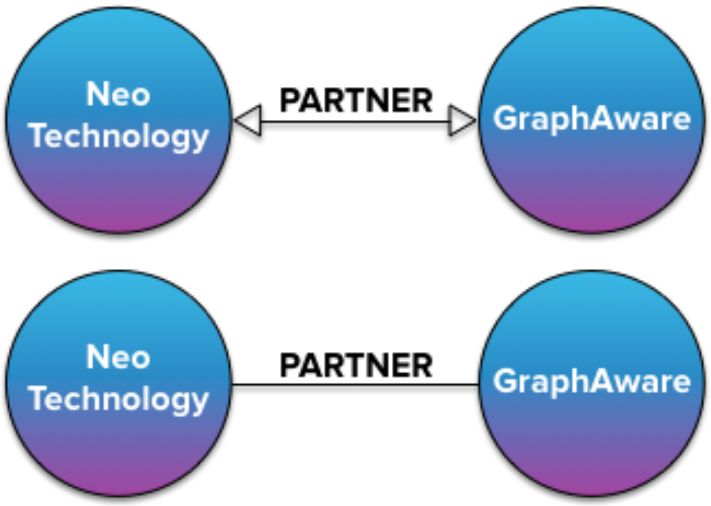


Dies ist ein Fehler, der vielen unterläuft, wenn sie ein Neo4j-Diagramm modellieren und in zwei Richtungen gehende Beziehungen definieren. Da eine Beziehung die zweite ausdrückt (symmetrische Beziehung), ist dies sowohl hinsichtlich des benötigten Platzes als auch des Graphen-Durchlaufs unwirtschaftlich. Neo4j erlaubt es einem, die Beziehung in beide Richtungen nachzuvollziehen, sprich auch in die gegengesetzte Richtung der Kanten. Dazu kommt, dass aufgrund der Art und Weise, wie Neo4j Daten speichert, die Geschwindigkeit des Durchlaufs nicht von ihrer Richtung abhängt.

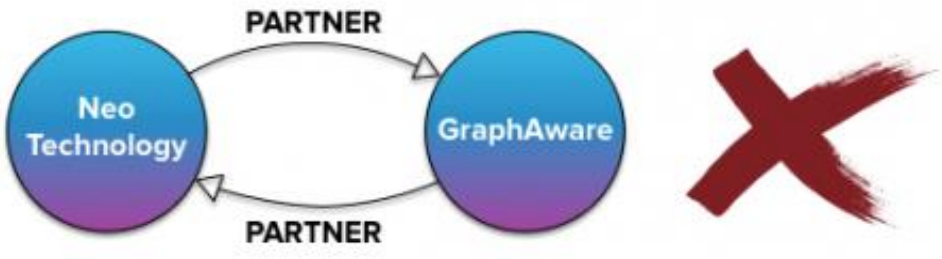
7.4. Zweiseitige Beziehungen

Einige Beziehungen sind natürlicherweise zweiseitig. Ein klassisches Beispiel ist Facebook oder eine Freundschaftsbeziehung. Die Beziehung beruht auf Gegenseitigkeit: wenn man mit jemandem befreundet ist, so ist auch dieser jemand (vielleicht) mit einem selbst befreundet. Abhängig davon, aus welcher Warte man das Diagramm-Modell betrachtet, kann man sagen, dass es entweder wechselseitig oder nicht ausgerichtet ist.

Ein Beispiel: GraphAware und NeoTechnology sind Partnerunternehmen. Da es sich um eine Beziehung handelt, kann diese entweder als zweiseitige oder nicht ausgerichtete Beziehung modelliert werden.



Dies ist jedoch in Neo4j nicht möglich: Hier muss jede Beziehung einen Ausgangs- und einen Abschlussknoten haben. Anfänger greifen häufig auf das folgende Modell zurück, welches genau dasselbe Problem hat wie das zuvor genannte Eishockey-Beispiel: Redundanz.



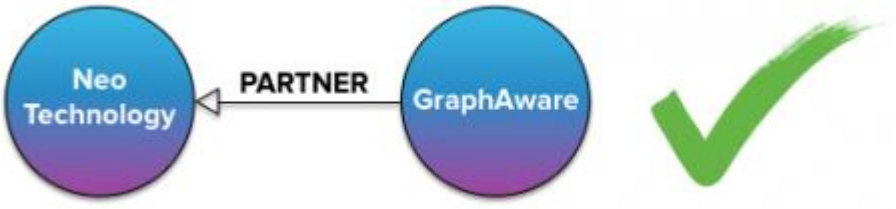
Die Neo4j-API erlaubt es Entwicklern, die Ausrichtung der Beziehung bei Bedarf komplett zu ignorieren, wenn Abfragen geschrieben werden. In Cypher könnte die Suche nach allen Partnerunter-nehmen von NeoTechnology zB wie folgt aussehen:

```
MATCH (neo) - [: PARTNER] - (partner)
```

Das Ergebnis wäre dasselbe, wenn man die Ergebnisse dieser zwei unterschiedlichen Abfragen zusammenführen würde:

```
MATCH (neo) - [: PARTNER] -> (partner) and MATCH (neo)
<- [: PARTNER]
```

Die richtige (oder zumindest effizienteste) Möglichkeit, Partnerschaftsbeziehungen zu modellieren ist daher, eine einzelne PARTNER-Beziehung zu verwenden, die beliebig ausgerichtet sein kann.



7.5. Zusammenfassung

Beziehungen in Neo4j können gleich schnell in beide Richtungen durchlaufen werden. Darüber hinaus kann die Ausrichtung völlig ignoriert werden. Aus diesem Grund ist es nicht notwendig, zwei verschiedene Beziehungen zwischen Entitäten zu erstellen, wenn die Kanten der gegengesetzten Richtung dieselbe Bedeutung haben.

7.6. Grafische Datenbankmodellierungsmethodologie

Im folgenden Artikel werden die Gestaltung, Implementierung und Überprüfung einer Neo4j Graphen-Datenbank für das Projekt „BestPartyToday“ gezeigt (hierbei handelt es sich um eine weltweite Teilliste, welche detaillierte Statistiken über Events anbietet). Aktuell verwendet die MySQL-Datenbank dutzende Millionen Datensätze, weshalb man auch darauf warten muss, dass Daten aus einem relationalen Datenbankspeicher in einen Graphen übertragen werden. Im heutigen Arbeitspapier wird allerdings nur die Graphen-Datenbank von Tabellen und Attributen modelliert, die aus der aktuellen Struktur der MySQL-Datenbank ausgewählt wurden.

Das konzeptionelle Design der Graphen-Datenbank wird auch als Whiteboard-freundlich bezeichnet. Um schnell alle Ideen und Erkenntnisse wiederzugeben, die im Zuge des Gestaltungs-Brainstormings gewonnen wurden, bedarf es nur eines Flipcharts. Das daraus resultierende Layout kann sehr leicht anderen Projektgruppen (Verkaufsabeilung, Kontomanager, Vertriebsmannschaft etc.) präsentiert werden, welche keine technische Ausbildung haben. Es ist nicht länger ein Problem für sie, das Design zu verstehen, da sich das Layout einfach an das reale Leben anpassen lässt.

7.7. Implementation des konzeptionellen Modells

Das Top 5-Diagramm der aktuellen relationalen Datenbank, repräsentiert durch Symbole, welche die Knoten und Richtungspfeile für Beziehungen darstellen, wird im schlussendlichen Graphendiagramm-Design des BestPartyToday-Projekts verwendet. Anhand der Entitäten (Knoten und Beziehungen) wird der Graph durchstöbert. Dabei werden die benötigten Daten gewonnen, welche anschließend verarbeitet und dem Benutzer der Anwendung angezeigt werden. Ihre Bedeutung ist in der nachfolgenden Tabelle beschrieben. Ein Überblick über die Diagramm-Modell-Entitäten.

8. NoSQL-Datenbanken

NoSQL-Datenbanken sind das Thema, dem sich diese Arbeit widmen wird. Zunächst einmal muss erläutert werden, was eine NoSQL-Datenbank ist. Im nächsten Kapitel geht es dann um das CAP-Theorem. Dieses erklärt, dass einer der Gründe dafür, warum es so viele NoSQL-Produkte gibt, jener ist, dass jedes davon bestimmte Eigenschaften nur auf Kosten von anderen zur Verfügung stellen kann. Aus diesem Grund muss sich der Benutzer entscheiden, welche Kombination von Eigenschaften für ihn die wichtigste ist. Auf dieser Grundlage fällt dann die Entscheidung für ein Produkt.

Hinzu kommt, dass sich die Arbeit damit auf die Skalierbarkeit fokussiert, sprich die Fähigkeit, die Leistung des Datenbankservers zu steigern. Dies gelingt entweder durch die Verbesserung des Servers mittels effizienterer Hardware oder durch Aufteilung der Da-

tenbank auf mehrere Server. Ähnlich wichtig wie die Skalierbarkeit ist auch das Teilen, eine Methode, mit der Datenbanken in mehrere Teile aufsplittet werden, die eigenständig arbeiten können und schneller sind als eine große Datenbank. Als nächstes geht es um die bekanntesten NoSQL-Produkte, wobei drei Vertreter davon herausgegriffen werden, auf die näher eingegangen wird.

8.1. Beschreibung

Wie bereits in der Einleitung erwähnt, befassen sich NoSQL-Datenbanken nicht mit der relationalen Art der Datensteuerung, sondern einer anderen. Sie basierend nicht primär auf Tabellen und verwenden kein SQL, um mit Daten zu arbeiten. Ihre Domäne ist eine hohe Suchoptimierung auf Kosten geringer Funktionalität, welche oft auf einfache Datenspeicherung begrenzt ist. Diese Mängel im Vergleich zu SQL werden durch Skalierbarkeit und Leistung in bestimmten Datenmodellen wettgemacht. NoSQL ist für die Speicherung großer Datenmengen geeignet, die nicht verknüpft werden müssen. Strukturierte Daten sind hingegen erlaubt.

NoSQL-Datenbanken sind in der Lage – mit Ausnahme von Konstanz – sämtliche ACID-Kriterien (siehe Kapitel 9.1) bei der Ausführung von Arbeitsvorgängen vollumfänglich zu unterstützen. In manchen Situationen erhält man aufgrund der Missachtung von ACID eine bessere Zugänglichkeit sowie bessere Skalierbarkeit. Dieser Zugang ohne „starke Konstanz“ ist für NoSQL geeignet, welche ihn auch oft anwendet. NoSQL hat eine dezentralisierte Architektur, welche fehlertolerant ist. Einige Daten können sich auf mehreren Servern befinden, weshalb der Ausfall eines Servers toleriert werden kann. Diese Datenbanken skalieren für gewöhnlich horizontal und veralten große Datenmengen, wobei Leistung wichtiger ist als Konstanz.

NoSQL ist buchstäblich eine Kombination aus zwei Wörtern: No und SQL. Ihre Technologie steht jener von SQL entgegen. Die Abkürzung mag etwas verwirrend sein und es gibt keinen Konsens unter den Menschen darüber, was es bedeutet. Die am weitesten verbreitete Auffassung ist jedoch, dass es sich um ein Akronym für „Not only SQL“ handelt. Was auch immer genau diese Abkürzung bedeutet, NoSQL ist mittlerweile ein Überbegriff für eine eigene Art von Datenbanken, welche nicht zu den bekannten RDBMS (relationalen Datenbank-Managementsysteme) gehören und gemeinhin mit großen Datenmengen assoziiert werden.

8.2. Was ist NoSQL?

Zunächst einmal ist es wichtig zu betonen, was NoSQL ist. Es ist definitiv KEIN SQL, weshalb es dazu tendiert, relationale Datenbanken abzulehnen. Die Abkürzung NoSQL bedeutet „Not Only SQL“, was so viel heißt wie dass es durchaus eine Alternative zu SQL gibt, die in bestimmten Fällen besser geeignet ist.

Bei größeren Anwendungen sieht man, dass für einige Daten relationale Datenbanken, für andere NoSQL-Datenbanken und für andere Daten wiederum andere NoSQL-Datenbanken geeignet sind. Dieser pragmatische Ansatz, mehrere Datenbanken in einem Projekt zu vermischen, wird of als Polyglott-Beharren bezeichnet.

NoSQL-Datenbanken entstehen (und entwickeln sich) als Lösungsansatz aus einem echten Problem heraus. Sie stellen eine durchaus praktikable Lösung dar und sind nicht nur ein dubioses Konstrukt, welches sich realitätsferne Akademiker in ihrem Kämmerchen zusammengesponnen haben. NoSQL-Datenbanken werden gemeinhin im Kontext von Projekten geboren, die mit großen Datenmengen arbeiten müssen: Facebook (Cassandra), Google (BigTable), Amazon (Dynamo), LinkedIN (Voldemort)

Die Verwendung einer NoSQL-Datenbank kann auch sinnvoll sein, wenn weniger Daten verfügbar sind (was eher die Domäne relationaler Datenbanken ist), da einige NoSQL-Datenbanken ein Datenmodell anbieten, das für gewisse Anwendungen natürlicher ist. Aber zurück zur einleitenden Frage: Eine NoSQL-Datenbank ist eine Datenerhaltungssoftware, welche eine Alternative zur klassischen relationalen Datenbank darstellt.

9. Transaktionen

Eine Transaktion ist eine logische Arbeitseinheit, die aus einem oder mehreren SQL-Statements besteht, welche atomar sind, was die die Erholung von SQL-Transaktionsfehlern anbelangt. Sie beginnt mit dem SQL-Bereitstellungsbefehl (SELECT, INSERT). Veränderungen, welche durch einen Arbeitsvorgang durchgeführt werden, sind für andere in Wettbewerb stehende Arbeitsvorgänge unsichtbar, es sei denn, die Transaktion endet nicht.

Ein Arbeitsvorgang kann auf vier verschiedene Arten enden:

- COMMIT beendet den Arbeitsvorgang erfolgreich und die Veränderungen werden permanent gespeichert.
- ROLLBACK unterbricht die Transaktion und alle Veränderungen werden aufgehoben, die Datenbank wird auf den Stand vor der Transaktion zurückgesetzt.
- Innerhalb des Programms: Endet das Programm erfolgreich, Arbeitsvorgang.
- Innerhalb des Programms: Endet das Programm mit einem Fehler, wird die SQL-Transaktion abgebrochen.

Das SQL-Datenbank-Zugriffsmanagement definiert zwei Befehle, um auf Tabellen zuzugreifen:

- GRANT und REVOKE. Der Sicherheitsmechanismus basiert auf Autorisierungsidentifikatoren – Inhaberverhältnis – Privilegien

9.1. Lösung: Transaktionen

Transaktionen können als die Reihenfolge von Datenbank-Operationen verstanden werden, welche die folgenden ACID-Kriterien erfüllen.

1. **Atomarität (Atomicity):** Es werden entweder alle Arbeitsvorgänge innerhalb einer Transaktion ausgeführt oder keine.

2. **Konstanz (Consistency):** Die Datenbank befindet sich sowohl vor als auch nach einem Arbeitsvorgang in einem gleichbleibenden Zustand. Alle Integrationseintschränkungen müssen erfüllt werden.

3. **Isolation:** Einzelne Transaktionen werden isoliert. Veränderungen, die während der Ausführung eines Arbeitsvorgangs gemacht werden, sind nicht in anderen Transaktionen sichtbar.

4. **Beständigkeit (Durability):** Nachdem der Arbeitsvorgang erfolgreich ausgeführt wurde, werden die Daten gespeichert und können nicht verloren gehen.

9.2. Arbeitsvorgänge in SQL

In SQL stehen die folgenden Befehle zur Verfügung, um Arbeitsvorgänge zu erledigen:

BEGIN	startet eine Transaktion ... befiehlt innerhalb einer Transaktion etwas ...
COMMIT	speichert und beendet die Transaktion
BEGIN	... befiehlt innerhalb einer Transaktion etwas ...
ROLLBACK	kehrt Veränderungen an einer Transaktion um und unterbindet diese

Isolationsstufen

ACID-Anforderungen sind ziemlich stark und zeitaufwendig. Deshalb können diese Bedingungen durch sogenannte Isolationsstufen reduziert werden. Es gibt vier Stufen (von der schwächsten bis hin zur stärksten):

1. READ UNCOMMITTED

Es ist einem Arbeitsvorgang möglich, die Daten zu lesen, welche von einer anderen Transaktion aufgezeichnet wurden, ohne dass besagte andere Transaktion mit COMMIT abgeschlossen wird. Das Lesen solcher Daten wird als „schmutziges Lesen“ (dirty read) bezeichnet. Daten, welche auf diese Weise gelesen werden, können inkonsistent sein (Zeit läuft von oben nach unten ab):

2. READ COMMITTED

Auf dieser Stufe der Isolation kann es nicht mehr zu „schmutzigem Lesen“ kommen. Die Daten, die hier ausgelesen werden, sind das Ergebnis einer erfolgreich mit COMMIT beendeten Transaktion. Was hingegen auftreten kann, ist ein reproduzierbarer Lesevorgang. Werden Daten in einer Transaktion mehrfach ausgelesen, kann es vorkommen, dass die wiederholt durchgeführten Lesevorgänge nicht immer zum selben Ergebnis führen. Es ist möglich, erfolgreich eine Transaktion durchzuführen, die zwischen den Transaktionsdatenauslesevorgängen einige Daten aufgezeichnet, bearbeitet oder gelöscht hat.

3. REPEATABLE READ

Auf dieser Stufe wird sichergestellt, dass es keine nicht reproduzierbare Lesevorgänge mehr gibt. Die Daten, die einmal ausgelesen wurden, können sich nicht verändern, wenn sie erneut gelesen werden. Allerdings kann ein Phantom-Lesevorgang auftreten. Wer-

den Daten in einer Transaktion mehrfach ausgelesen, kann es vorkommen, dass die Ergebnisse des Lesevorgangs neue Zeilen enthalten, die in den vorangegangenen Lesevorgängen nicht aufscheinen. Besagte Transaktionsdaten lesen erfolgreich eine simultan durchgeführte Transaktion aus, welche neue Daten produziert hat.

4. SERIALIZABLE

Auf dieser Stufe kann es zu keinem der oben beschriebenen Phänomene kommen, da zwangsweise die ACID-Kriterien eingehalten werden.

10. Verfahren und Funktionen, Trigger und Sequenzen

Wodurch unterscheiden sich Funktionen, Methoden und Verfahren?

Alle drei (Funktionen, Methoden und Verfahren) sind ziemlich ähnlich: Es handelt sich um eine DEFINIERTE Reihenfolge von Befehlen, den Teil eines Programms, der wiederholt von anderen Teilen des Programms abgerufen werden kann.

10.1. Formelle Ansicht

Aus einem formellen Blickwinkel folgt man der Nomenklatur der vorgegebenen Programmiersprache.

- Funktion ist ein Begriff funktionalen Programmierens, welcher in Sprachen wie JavaScript oder Haskell auftritt.
- Wenn man von Methoden spricht, bezieht man sich auf Funktionen des Objekt-Orientierten Programmierens (OOP), wo die Datenstrukturen und funktionale Codes mit Objekten verknüpft sind. Man findet diese Methoden in Java, Smalltalk und anderen Sprachen.
- Verfahren ist ein Begriff, den man aus verfahrenstechnischen Sprachen kennt. Man findet sie z.B. in Pascal, aber auch in ein paar Datenbanksystemen, sogenannten gespeicherten Verfahren.

Man kann Funktionen, Methoden und Verfahren jedoch auch aus einem physikalischen Blickwinkel (gemäß ihrer Charakteristika und Bedeutung) betrachten und von der Terminologie einer speziellen Programmiersprache abweichen.

Funktion
Funktionen in der Programmierung sind jenen in der Mathematik sehr ähnlich. Eine Funktion hat ein bestimmtes Definitionsfeld (Art der Eingabeparameter) und eine bestimmte Bandbreite an Werten (Art der wiederzugebenden Werte).

Beispiel eines einfachen JavaScript-Merkmals:

```
Function sum (a, b) {return a + b};
```

Methode

In Objektsprachen wie Java gibt es Methoden anstelle von Funktionen, was jedoch nicht deren Eingabe verhindert.

```
Public static int sum (int a, int b) {return a + b};
```

Obwohl *sum ()* formell eine Methode (öffentlich und statisch) ist, handelt es sich Wahrheit um eine Funktion und die Klasse hier erfüllt nur die Rolle eines Namensraums (zusammen mit dem Namen des Pakets). Es wird jedoch keine Instanz geschaffen (oder nicht). Es handelt sich um eine Gestaltungsmusterbibliothek-Klasse (Design Pattern Library Class).

In Java findet man solche Funktionen, z.B. in der java.lang.Math-Klasse. Die folgende Funktion gibt die größerer von zwei Zahlen zurück:

```
Int x = java.lang.Math.max (a, b);
```

Verfahren

Das Verfahren ist ein spezieller Anwendungsfall einer Funktion: Es hat keinen Rückgabewert und muss auch nicht zwangsläufig Eingabeparameter haben. Verfahren werden oft bei der Stapelverarbeitung verwendet, zB wird jede Stunde ein Verfahren abgerufen, welches die in der Datenbank akkumulierten Bestellungen verarbeitet und diese an ein anderes System weiterleitet.

Nachfolgendes Beispiel zeigt ein sehr einfaches PostgreSQL-Verfahren, welches die Werte der Spalten A und B in nicht adressierten Zeilen summiert und das Ergebnis in der Spalte c speichert:

```
CREATE OR REPLACE FUNCTION add ()
RETURNS void AS
$ BODY $
UPDATE table of summaries
SET c = a + b
WHERE c is NULL
$ BODY $
LANGUAGE sql VOLATILE;
```

Dieses Verfahren ist in SQL geschrieben. Darüber hinaus kann es in PLPGSQL geschrieben werden, welches die Entwicklung äußerst komplexer Verfahren ermöglicht. Alternativ kann auch die Sprache C oder eine andere verwendet werden.

Das oben gezeigte Verfahren (formell Funktion) hat keine Eingabeparameter oder Rückgabewerte, es ist einfach eine Reihenfolge von Sprachbefehlen, welche definiert und gespeichert wurden.

Es ist interessant, dass Verfahren Parameter haben können: Diese gibt es nicht nur im Kontext der klassischen Eingabe, sondern auch der Ausgabe (oder beiden). Das Ausgabeparameter-Verfahren ist ähnlich zur Funktion, obwohl es keinen klassischen Rückgabewert hat. So können zB Daten an einen Prozessor weitergegeben werden, welcher diese modifiziert.

Diese Verfahren können auch in Java simuliert werden. Hier ist ein Beispiel für eine Struktur:

```
Public class Person {public String name;
Public String Surname; Public String fullname};
```

Ein Verfahren:

```
Public class StoredProcedures {public void reportNameName
(Person o) {o.celeName = o.name + ”” + oName}};
```

Die Daten (sprich die Person) werden an das Verfahren weitergegeben, außerdem werden die erforderlichen Anpassungen vorgenommen. Wenn man diesen Stil allerdings in Java programmiert, macht man wahrscheinlich etwas falsch und macht sich bereit für die Hauptvorteile von PPE.

Merke:

Vorsicht vor Werttransfers! Wenn man innerhalb des Java-Verfahrens eine neue Person einer Variable zuordnet, wird diese Veränderung nicht im Verfahren abgebildet. Die ursprüngliche Person bleibt unangetastet. Man kann jedoch mit den Attributen der ursprünglichen Person arbeiten – in diesem Fall dem Code des Verfahrens und dem Code der die Instanz derselben Person „sieht“.

Methode

Methoden sind Teil der Klasse und (wenn sie nicht statisch sind) eng verwandt mit der vorgegebenen Klasseninstanz (Objekt). Deshalb sind sie keine Funktionen, welche mit globalen Variablen und Daten arbeiten, aber sie haben Zugriff auf die Variablen einer gegebenen Instanz (in diesem Fall Individuen).

Das vorherige Beispiel kann „objektiver" wie folgt überschrieben werden:

```
Public class Person {public String name; Public
String Surname; Public String getCelÉName ()
{return name + ”” + surname}};
```

Es ist keine gespeicherte Verfahrensklasse notwendig, um die benötigte Funktionalität näher an die Daten heranzurücken (zur Personenklasse). Man muss an dieser Stelle anmerken, dass sich nicht einmal der errechnete Wert des ganzen Namens speichern lässt – sprich er wird nur dann errechnet, wenn ihn jemand benötigt, z.B. durch den Abruf der *getNameName () Methode.*

Was ist ein Trigger?

Ein Trigger ist ein Verfahren, das automatisch ausgelöst wird, wenn etwas passiert. Er lässt sich sowohl für DML (Modifikations-) als auch für DDL (Erstellungs-) Operationen festlegen. Überdies lässt sich festlegen, ob er vor (BEFORE) oder nach (AFTER) einer Operation ausgelöst werden soll. Es ist interessant, dass der Trigger als Tabelle bezeichnet (aber nicht vorgeschlagen) werden kann und dass eine beliebige Anzahl an Triggern für dieselbe Tabelle und dasselbe Ereignis definiert werden kann.

11. Analytische Werkzeuge - OLAP

11.1. OLAP – Informationen unter Kontrolle

Welche Kunden bringen 80% des Profits? Wie hat sich der Erfolg, Kunden zu halten, nach dem Loyalitäts-Programm verändert? Wie wirken sich verschiedene Business-Events (Kampagnen, Einführung eines neuen Produkts, Veränderungen in Unternehmensprozessen usw.) und der Markt (Wettbewerberaktivitäten, Veränderungen der wirtschaftlichen Rahmenbedingungen usw.) auf die Verkaufsfluktuation aus? Wie sieht der Vergleich mit der letzten Periode aus? Diese und ähnliche Fragen werden von jedem realistischen Manager gestellt. Jeden Tag versucht er, das Verhalten des Unternehmens auf Basis der Analyse ihm zur Verfügung stehender Daten zu verstehen. In gut laufenden Unternehmen haben ausgewählte Abteilungen Analytiker, welche nur diese Aufgabe übernehmen und nur begründete Empfehlungen an Manager abgeben. Alle diese Personen brauchen eine fertige Infrastruktur und Werkzeuge, welche es ihnen ermöglichen, ihre wichtige Aufgabe zu erfüllen.

11.2. Was die Arbeit erleichtert und wie

Der Weg zu erfolgreicher Datenanalyse beginnt mit deren Vorbereitung. Der erste wichtige Schritt ist Daten aus Unternehmenssystemen in eine geeignete Analyse- und Berichterstattungsform zu bringen. Bereits während dieser Vorbereitung ist es notwendig, sicherzustellen, dass die verschiedenen Daten und Datenqualitäten korrekt angeglichen werden. Es muss jede Verknüpfung zum Quellensystem zuverlässig dokumentiert werden, sodass die Ergebnisse immer glaubwürdig und verifizierbar sind – um Fehler zu beheben, welche in den ursprünglichen Daten auftreten können. Die daraus resultierten Daten werden in einer relationalen Datenbank gespeichert, wo sie im für operationale Analysen notwendigen Umfang verfügbar sind. Sofort verwendbare Daten setzt man ein, um detaillierte Berichte über die Leistungsfähigkeit eines Unternehmens und einfachere Analysen basierend auf detaillierten Daten zu erstellen. Für das folgende Beispiel allerdings benötigt man eine andere Art der Information: Eine landesweit aktive Firma verkauft verschiedene Produktkategorien. Der Manager unterteilt individuelle Branchen in Regionen. Die verkaufte Produktkategorie hat drei Ebenen: Kategorien, Unterkategorien, spezifisches Produkt. Die Firma hat eine logische Definition, der zufolge Verkäufe durch eine Planung geregelt sind. Die echte Einhaltung des Plans wird durch die Summe aller Produkte und aller Filialen gewährleistet.

11.3. Wie erhält man diese Informationen?

Zu diesem Zweck werden fortgeschrittene Analysen als Multidimensional bezeichnet. Der Manager kann messbare Geschäftsparameter in Form von Zusammenhängen, Blickwinkeln und auf verschieden detaillierten Ebenen verfolgen. Eine solche Analyse erfordert OLAP-(on-line analytical processing) Technologie, um sicherzustellen, dass Daten auf jene Art und Weise gespeichert und vorberechnet werden, dass spätere Abfragen für eine sinnvolle Dauer währen. Die OLAP-Technologie erlaubt es, mit Daten auf einer zusammenfassenden Ebene zu arbeiten, ein Problem oder ein Interessensgebiet zu identifizieren und die nächste Detailstufe zu erreichen, die im Zusammenhang der Entscheidungsfindung von Relevanz ist. Ein typisches Gebiet, in der die Vorteile von OLAP sichtbar werden, ist in der Verkaufsanalyse.

Abbildung 1: OLAP in der Oracle Business Intelligence – Discoverer

Mithilfe von OLAP-Werkzeugen kann ein Manager kontrollieren, wie sich die Realität im Widerspruch zur Planung entwickelt. Werden die Erwartungen nicht erfüllt, zeigt die Trendkurve diese Information rechtzeitig an, wodurch verhindert wird, dass das Geschäftsjahr mit einem Fiasko endet. Aufsummierte Zahlen lassen sich leicht herunterbrechen, sodass man sich eine Zusammenfassung der Vorgaben für jede Region anschauen kann. Eine Analyse kann so z.B. zeigen, dass das Problem nur eine Region betrifft und die anderen Regionen voll auf Plan sind. Durch weiteres Hineinzoomen gelangt der Manager auf die Filialebene, auf welcher sich identifizieren lässt, welche Filialen in der problematischen Region deutlich hinter den Erwartungen liegen. Das System erlaubt

ein Aufteilen der Zahlen bis hin zur Kategorien- und Subkategorienebene, bei Einzelprodukten können die Zahlen jedoch nur kumuliert und nicht nach Verkaufserfolg pro Standort ausgewiesen werden.

Basierend auf diesen Informationen ist es dann möglich, korrektive Maßnahmen zu entwickeln und umzusetzen, um die Situation zu verbessern. Das Beispiel zeigt, dass OLAP-Daten tatsächlich in einer Baumstruktur gespeichert werden, sprich in einem Würfel. Auf jeder Tiefenstufe dieses Baums werden Werte für den Unterbaum berechnet. So eine hierarchische Struktur lässt sich auf Grundlage der meisten Unternehmensdaten festlegen. Diese trägt wesentlich zu einer Vereinfachung der Daten bei und erleichtert die Analyse, wenn die Daten ausgewertet werden. Der Preis für diese Vereinfachung sind die Zeit und der Platz, welche zur Berechnung und zur Speicherung des Baums benötigt werden. Wenn der der Würfel allerdings gefüllt ist, kann er adäquat sehr komplizierte Abfragen beantworten. Dies betrifft vor allem das Vergleichen von Zeitfenstern, welche durch komplexe Bedingungen eingeschränkt sind.

Ein Beispiel für eine typische, komplizierte Anfrage ist: „Welche zehn Kunden aus den 20% unserer am wenigsten profitablen Bezirke sind im Vergleich zum letzten Jahr für den größten Umsatzanstieg in unserem Unternehmen verantwortlich?“ Das Ergebnis kann als Anlass dafür genutzt werden, diesen Kunden besondere Aufmerksamkeit zu widmen. Mit OLAP ist es möglich, Daten unter verschiedenen Gesichtspunkten – vielen Dimensionen – abzufragen und diese Gesichtspunkte mit beliebigen Kennziffern zu versehen. Wie bereits gezeigt wurde, ist es nicht ratsam, Daten für diese Art von Abfragen in relationaler Form abzuspeichern.

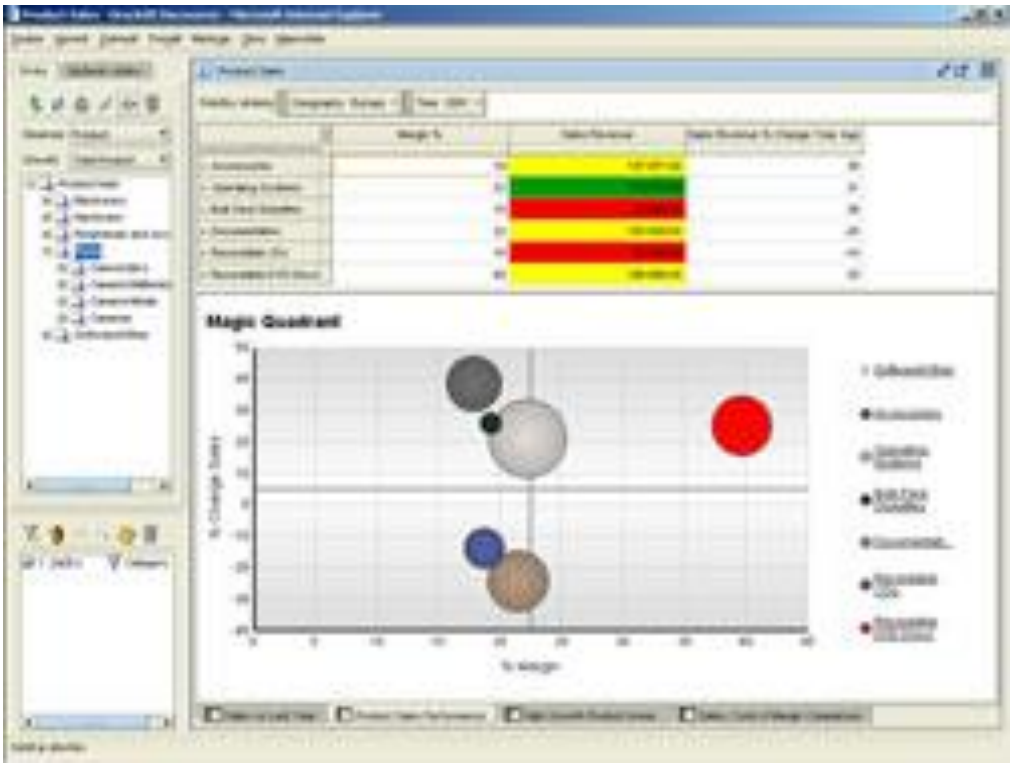


Abbildung 2: Verkaufsanalyse in der Oracle Business Intelligence – Discoverer

11.4. Analytische Instrumente

Bis jetzt wurden Wege und Technologien der Datenspeicherung beschrieben. Die Arbeit mit diesen – sprich die Durchführung von Analysen – erfordert geeignete Werkzeuge. Die Auswahl der richtigen analytischen Werkzeuge ist der Schlüssel zur effektiven Sammlung von Informationen. Es ist dabei besonders wichtig, die Expertise von Analytikern zu berücksichtigen, welche bereits mit den ihnen zur Verfügung stehenden Ressourcen arbeiten. Sehr häufig werden in diesem Kontext Werkzeuge aus der Microsoft Office-Familie verwendet. Aus diesem Grund ist es wichtig, dass die ausgewählten Lösungen im Zusammenspiel mit den bisher genutzten Werkzeugen funktionieren. Dies führt zu geringeren Kosten, wenn bei der Arbeit mit Informationen ein effizienterer Weg eingeschlagen wird. Gleichzeitig müssen der sichere Zugang zu Informationen, die automatisierte Verteilung von Berichten, die qualitativ hochwertige, grafische aufbereitete Datenausgaben für Präsentationen sowie bei Bedarf die Integration in das Unternehmensintranet sichergestellt werden.

Die Hauptakteure sind

Bei der ganzen Flut an Technologie darf man nicht auf die Menschen vergessen. Ein paar der wertvollsten Mitarbeiter, die ein Unternehmen gewinnen kann, sind erfahrene Analytiker. Nur solche erfahrenen Analytiker sind in der Lage, aus den Unmengen an Daten das herauszulesen, worauf es ankommt und zielgerichtete Empfehlungen für das Management zu formulieren.

11.5. Der OLAP-Würfel

Der OLAP-Würfel ist eine Methode, Daten zu organisieren, welche über eine zweidimensionale Tabellenanordnung hinausgehen, weshalb jede Datendimension auf einer Achse des Würfels gespeichert wird. Damit werden ein paar der Einschränkungen relationaler Datenbanken überwunden.

Das Anordnen von Daten in Würfelvektoren ermöglicht den Zugriff darauf aus verschiedenen Perspektiven (Dimensionen), aber auf dieselbe Weise. Die ausgeführte Systemleistung macht es schwierig, viele RDBMS-Tabellen miteinander zu kombinieren. Allerdings erlaubt die physische Datenspeicherung in Würfeln kein schnelles Bearbeiten, weshalb der gesamte Würfel wieder aufbereitet werden muss. Der Würfel besteht aus Werten, welche in Dimensionen kategorisiert sind. Die Struktur wird mithilfe relationaler Tabellen in ein Sterndiagramm oder Schneeflocken-Schema implementiert. Es handelt sich in der Regel um eine Eltern-Kind-Struktur, wo die Eltern-Elemente die Verdichtung des Ursprungs darstellen und sie selbst in ihren Eltern-Elementen kumuliert werden können.

11.6. Grundlegende Datenwürfeloperationen

Da sie aus analytischer Sicht notwendig sind, werden die nachfolgenden Datenwürfeloperationen durchgeführt, um die Verarbeitung und Projektion von Daten sowie deren Verständnis zu erleichtern:

Slice a Cube: Begrenzung einer oder mehrerer Dimensionen auf eine Untermenge eines Elements.

Cubming: Begrenzung einer oder mehrerer Dimensionen auf eine Untermenge zweier oder mehrerer Elemente.

Roll up and drill down: Nach oben und nach unten durch die Datenhierarchie navigieren.

Pivot: Rotieren des Würfels, um Datenbeziehungen aus verschiedenen Blickwinkeln zu betrachten.

Aggregation: Verdichtung gemäß der Beziehungen, welche von Formeln vorgegeben wurden.

12. Spezifika von Datenbanksystemen – Technologien für den Zugriff auf Datenbanken – Geographische Informationssysteme

Eine Datenbank ist ein Dateisystem mit einer festgelegten Datensatzstruktur. Diese Dateien sind mithilfe von Schlüsseln miteinander verbunden. Im weiteren Sinne enthält eine Datenbank Software-Ressourcen, welche die Manipulation von und den Zugriff auf gespeicherte Daten ermöglichen. Diese Software wird in Tschechiens technischer Literatur als Datenbank-Managementsystem (DBMS) bezeichnet. Für gewöhnlich wird die Bezeichnung Datenbank – abhängig vom jeweiligen Kontext – entweder mit gespeicherten Daten oder mit der Software (DBMS) gleichgesetzt.

12.1. Überblick über den Daten Zugriff in der ASP.NET-Technologie

Visual Studio 2010

Web-Anwendungen greifen für gewöhnlich auf Datenquellen zu, um dort dynamische Daten zu speichern bzw. Diese von dort abzurufen. Man kann einen Datenzugriffscode schreiben, indem man die Klassen im System.Data Namensraum (in der Regel als A-DO.NET bezeichnet) und im System.Xml Namensraum verwendet. Dieser Zugang war in den Vorgänger-Versionen von ASP.NET üblich.

Darüber hinaus erlaubt es die ASP.NET-Technologie, Datenverknüpfungen deklarierend zu erklären. In den meisten gebräuchlichen Szenarios ist kein Code erforderlich. Dies können die folgenden sein:

- Auswahl und Anzeige von Daten
- Sortieren und Paginieren von Daten sowie deren Aufnahme in den Cache-Speicher
- Aktualisieren, Einfügen und Löschen von Daten
- Datenfilterung mithilfe von Laufzeit-Parametern
- Erstellen von Master-Detail-Szenarios mithilfe von Parametern

ASP.NET enthält einige Arten von Server-Steuerungen, welche Teil des deklarativen Datenbindungs-Modells sind. Dazu gehören Datenquellensteuerungen, Datenbindungssteuerungen und eine erweiterte Abfragesteuerung. Diese Steuerungen bewältigen die grundlegenden Aufgaben, welche von einem unausgereiften Seitenmodell benötigt werden, um Daten auf einer ASP.NET-Webseite anzuzeigen und zu aktualisieren. Steuerun-

gen ermöglichen es, der Webseite eine Datenbindungs-funktionalität hinzuzufügen, ohne die Lebenszyklusdetails der Seite verstehen zu müssen.

13. Datenbankobjekte

Der Begriff „Datenbank“ wird oft vereinfacht und als Synonym für ein Datenbanksystem (Datenbank-Maschine) oder ein Datenbank-Managementsystem gebraucht. Sie enthält nicht nur Kalkulations-tabellen, welche nur eine von vielen sogenannten Datenbank-Objekten (manchmal auch Datenbank-Entitäten) sind. Erweiterte Datenbankensysteme enthalten:

- **Ansichten oder Aussagen:** SQL-Befehle, die im Datenbanksystem bestimmt und gespeichert sind. Man kann diese (durch Anwendung des SELECT-Statements) auch in anderen Tabellen aufrufen.

- **Indizes für jede Tabelle:** Schlüssel werden oberhalb einzelner Tabellenspalten definiert (ein Schlüssel kann mehrere enthalten). Sie haben die Funktion, doppelte Einträge in Spalten der Datensätze zu behalten, welche über LUT-Tabellen (Nachschlagetabellen) für die Volltext-Suche definiert wurden.

- **Trigger:** Mechanismus für einzelne Zeilen einer Tabelle (oder die Tabelle selbst), welcher nach dem Verändern, Löschen oder Hinzufügen einer Zeile, dem Löschen einer Tabelle und einer vorprogrammierten Maßnahme abgerufen wird (zB zur Überprüfung der Datenintegrität).

- **Benutzerdefinierte Verfahren und Funktionen:** Einige Datenbankmaschinen unterstützen die Speicherung bestimmter Code-Teile, welche eine bestimmte Reihenfolge von Befehlen (Verfahren) in einer Datenbank unter Verwendung vorgegebener Tabellen ausführen oder Ergebnisse zurückgeben (Benutzerfunktionen). Sie können Parameter haben, die sich im Wesentlichen in Eingabe (IN), Ausgabe (OUT) sowie Ein- und Ausgabe (INOUT) gliedern.

- **Ereignisse, auch („cash“):** Hierbei handelt es sich um Verfahren, welche an einem bestimmten (vom Benutzer vorgegebenen) Datum zu einer bestimmten Zeit bzw. wiederholt inner-halb einer festgelegten Periode ausgelöst werden. Sie dienen zur Wartung, temporären Datenschmierung oder zur Überprüfung der Verweisintegrität.

- **Formulare:** Einige Datenbanksysteme wie Microsoft Acces erlauben es Benutzern, Eingabe-formulare zu erstellen, die optisch zur Eingabe einladen. So kann ein Benutzer zB das Layout jedes Eingabefelds einer vorgegebenen Tabelle, Etikette usw. festlegen.

- **Berichte:** Ähnlich wie Berichtsformulare erlauben sie es den Benutzern, das Layout für die Felder vorgegebener Tabellen festzulegen, welchen aktuelle Werte hinzugefügt werden. Sie werden zur Datenausgabe (Druckversion, Präsentation oder einfache Ansicht) verwendet. Aufstellungen können ergänzt werden, zB

durch Filter, welche nur die gewünschten Datensätze herausfiltern.

- **Benutzerbefugnisse:** Bessere Datenbanksystemen stellen in der Regel Optionen zur gestaffelten Einschränkung individueller Zugriffsrechte auf Datenbankobjekte zur Verfügung. Es gibt dutzende Möglichkeiten detaillierter Einschränkungen, welche bis hin zur Festlegung einzelner Befehlsarten geht, die ein Benutzer verwenden darf.

- **Partitionierung:** Eine Möglichkeit, die Daten in einer Tabelle auf mehrere Festplatten aufzuteilen und dadurch die Last zu reduzieren, die sie zu tragen hat.

- **Prozesse:** Datenbankmaschinen können einen Überblick über die Prozesse geben, welche deren Dienste gerade verwenden.

- **Variable Einstellungen:** Typischerweise dutzende Variablen, die sich neu gestalten lassen und dadurch das Verhalten und die Leistung der Datenbankmaschine selbst beeinflussen.

- **Kollation:** MySQL bietet fortgeschrittene Optionen zum Erstellen einiger Dutzend Zeichen-Sets und Vergleiche, welche sich unter dem Sammelbegriff Kollation zusammenfassen las-sen. Solche Kollationen können auf der Ebene einzelner Textspalten, ganzer Tabellen oder ganzer Datenbanken eingerichtet werden (mit kaskadenförmiger Vererbung). Eine Kollation beeinflusst auch die Sortierung, zB der Wert utf8_czech_ci stellt sicher, dass korrekt nach dem Tschechischen Alphabet (sprich inklusive aller diakritischer Zeichen und CH) sortiert wird.

- **Visuelles E-R Schema:** (im MySQL INFORMATION.SCHEMA). Visuelle Darstellung von Beziehungen (s) voneinander abhängiger Felder (Fremdschlüssel) zwischen Tabellen.

13.1. Grundlegende Konzepte der Datenbanktechnologie

Einführung die Datenbanktechnologien

Mit dem Aufkommen der Microcomputer und ihrem Einzug in das tägliche Leben scheint ein gesteigertes Interesse an rechnergenerierten Daten einhergegangen zu sein. Dieses bezieht sich nicht nur auf Berechnungen, sondern auch auf die Produktion von Wissen. Diese Erklärung lässt sich sehr gut nachvollziehen, wenn man bedenkt, dass sogar die anspruchsvollsten Computerspiele mit der Zeit die Spieler ermüden und dass Programmieren im Wesentlichen ein Weg ist, die Lösungsart für ein System aus zwei Gleichungen mit zwei Unbekannten zu berechnen.

Jemand möchte sich gerne Informationen organisieren bzw. nach Informationen suchen, die er häufig verwendet und während der Verwendung verändert. Geeignete Mittel zur Speicherung solcher Informationen in Datenform vorausgesetzt, wäre dieser jemand gewillt, im Rahmen einiger Programmabende zu versuchen, relevante Daten zu produzieren, zu kaufen oder mit einem anderen vergleichbaren Datenenthusiasten auszutauschen. Ein Beispiel könnte eine Bibliothek, ein Büro, eine Flugticketbearbeitung, ein Stadtmuseum, ein Krankenhaus oder ein Geschäft sein.

Das fertige Objekt der beschriebenen Art kann dann als Informationssystem (IS) bezeichnet werden. Dieses IS ist eine Menge an Elementen in den wechselseitigen Beziehungen zwischen Information und Verfahren (Informationsverfahren), welche Daten verarbeiten und das Kommunizieren von Informationen zwischen Elementen sicherstellen. Aus Sicht eines Benutzers sollte das IS Konstruktionsmerkmale haben, welche die Manipulation von Daten (Dateneingabe, Abfrageformulierung, Einsatz von Anwendungsprogrammen) einfacher macht und gleichzeitig sicherstellt, dass die zur Verfügung gestellten Funktionen stark und schnell genug sind. Ein flexibles „leeres“ IS, welches heutzutage einem Benutzer bereitgestellt wird, kann sehr schnell sein, aber über einen ausreichen-den Wissensstand verfügen.

Das Ziel wird es sein, die sogenannte Datenbankverarbeitungstechnologie zu demonstrieren. Eine Datenbanktechnologie entspricht einer Menge von Konzepten, Ressourcen und Methoden, die verwendet werden, um Informationssysteme (IS) zu erschaffen. Auf der Grundstufe lässt sich die IS-Architektur mit der Datenbank wie folgt abbilden: Daten sind in einer Datenbank (DB) organisiert und werden von einem Programmpaket, welches Datenbank-Managementsystem (DBMS) genannt wird, verwaltet. Die Datenbank und das Datenbank-Managementsystem bilden zusammen das sogenannte Datenbanksystem (DBS). In Übereinstimmung mit der Terminologie lässt es sich auch wie folgt schreiben: DBS = DB + SBBD. Das IS verwendet die vom DBS bereitgestellten Daten entweder direkt oder verarbeitet diese mithilfe von Anwendungsprogrammen.

13.2. Geografische Informationssysteme

Ein geografisches Informationssystem (GIS) erlaubt die Speicherung, Verwaltung und Analyse von Gebietsdaten.

Die meisten realen Objekte und Phänomene treten auf irgendeiner Stelle der Erdoberfläche auf (z.B. Bäume, Häuser, Flüsse etc.) oder haben eine Beziehung zu einem bestimmten Platz auf der Erdober-fläche (ein Staatsbürger hat einen permanenten Wohnsitz, das Produkt wurde in einer bestimmten Fabrik hergestellt). Gleichzeitig treten diese Objekte zusammen mit anderen Objekten in diesem Raum auf, wo sie miteinander interagieren. Aus diesem Grund ist die Kenntnis eines Orts und der räumlichen Beziehungen zwischen Objekten sehr wichtig und kann eine wichtige Rolle im Hinblick auf zahlreiche menschliche Aktivitäten spielen. Diese können von der Gestaltung eines Atomkraft-

werks bis hin zur Gestaltung eines geschäftlichen Netzwerks und dessen Erfolgsauswertung reichen.

In der Praxis bedeutet dies, dass diese beiden Informationen, sprich die tatsächlichen Daten des Objekts und dessen Standort, gleichzeitig auf dem Computer gespeichert werden müssen. Diese Art von Daten bezeichnet man als geografische (oder räumliche) Daten. Ein Computersystem, welches das Speichern und Verwenden solcher Daten ermöglicht, nennt man geografisches Informationssystem.

Grundlegende Komponenten eines geografischen Informationssystems

Vít Voženílek beschreibt diese in seinem Buch Geographical Information System I. The Concept, History, Basic Components wie folgt:

- Hardware
- Software
- Daten
- Bedienungspersonal

Um Systeme effizient bedienen zu können, ist deren Balance essentiell. Die Hardware, oder auch technische Ausrüstung, bildet die technische Grundlage geografischer Informationssysteme. Die Soft-ware ist eine Menge von Programmen, welche alle Systemoperationen ausführen. Die Daten sind das Schlüsselelement jedes geografischen Informationssystems. GIS ist aus der Perspektive organisa-tioneller Strukturen ein wahrhaftiges System. Seine Tätigkeit ist die Sammlung von Aktivitäten, welche die Funktionen des Systems zur Verfügung stellen.

13.3. Geografische Daten

Dimensionen geografischer Objekte

Die grundlegende Unterteilung geografischer Objekte erfolgt nach der Anzahl der Dimensionen. Reale Objekte auf der Erdoberfläche sind immer dreidimensional. Um in das GIS übertragen werden zu können, müssen sie zuvor jedoch auf ein erforderliches Maß abstrahiert werden.

- 0D Geo-Objekte: dimensionslose Objekte, Punkte werden nur durch ihren Standort definiert. Ein Beispiel hierfür ist eine Bushaltestelle in einem GIS, welches Transport modelliert oder der GSM-Transmitter des GIS eines Mobilfunkanbieters, welcher Signalabdeckungen modelliert.

- 1D Geo-Objekte: eindimensionales Objekt, mittels Strichen abgebildete Streckenabschnitte (Kanten, Linien), begrenzte Länge und keine Fläche. Beispiele hierfür

sind Straßen und Flüsse.

- 2D Geo-Objekte: zweidimensional, Polygone, mit begrenztem Umfang, begrenzter Oberfläche.

- 3D Geo-Objekte: dreidimensionale Objekte, geometrischer Körper. In einem GIS werden sie nur in Ausnahmefällen verwendet. Die dritte Dimension wird im GIS am häufigsten mithilfe eines sogenannten Terrainmodels (DMT, DEM) an der Oberfläche modelliert. Hierzu wird es mit topologischen Oberflächen (2.5D) verknüpft.

Die Grundlage für jedes GIS bilden geografische Daten. Geografische Daten sind Informationen über die Erdoberfläche und die darauf befindlichen Objekte. Diese Daten können in Form einer bestimmten Informationsschicht dargestellt werden, welche auch Thema genannt wird. Jedes Thema repräsentiert ein bestimmtes Element (zB Straßen, Seen, Städte, etc.) Geografische Daten können in einem GIS mittels der folgenden zwei Basismodelle organisiert werden.

13.4. Datensammlung

Das Erhalten von Daten und das Verschieben von Informationen in das System erfordert eine Menge Zeit. Es gibt eine Anzahl von Methoden, die verwendet werden, um GIS-Daten einzugeben, welche in digitaler Form gespeichert sind. Ausgangsdaten, welche auf Papier ausgedruckt oder als PET-Film vorliegen, können digitalisiert oder eingescannt werden, um digitale Daten zu produzieren. Der Digitalisierer produziert Vektordaten als Punkte, Linien und Polygone. Auch das Zerlegen von Karten in das Format eines Rasters, der weiterverarbeitet werden könnte, ist geeignet, um Vektordaten zu produzieren.
Gesammelte Daten erhalten dank einer Koordinationsgeometrie genannten Methode direkten Zugang zum GIS. Standorte des Globalen Navigationssatellitensystem (GNSS) können auch gesammelt und anschließend in das GIS importiert werden. Aktuell gibt es einen Trend in der Daten-sammlung, der es Benutzern erlaubt, Computer im Feld zu benutzen, welche die die Möglichkeit bieten, Daten in Echtzeit über eine Internetverbindung oder sogar ohne Internetverbindung zu bearbeiten. Dadurch besteht kein Bedarf mehr daran, Bürodaten zu importieren und zu aktualisieren, nach diese in Feldarbeit gesammelt worden sind.

Dies schließt die Fähigkeit ein, Standorte miteinzubeziehen, welche mittels eines Laserentfernungs-messers gewonnen wurden. Neue Technologien ermöglichen es Benutzern, sowohl Karten zu erstellen als auch Feldanalysen durchzuführen, Projekte effizienter zu gestalten und Karten genauer zu zeichnen. Aus der Ferne erhobene Daten spielen auch eine wichtige Rolle bei der Datensammlung und werden mittels Sensoren gesammelt, die mit einer Plattform verbunden sind. Diese Sensoren können Kameras, digitale Scanner und LIDAR sein, während es sich bei den Plattformen zumeist um Luftfahrzeuge und

Satelliten handelt. Mitte der 1990er-Jahre machten in England Hybrid-Ballone, welche Helikites genannte wurden, das erste Mal Werbung für den Einsatz kompakter Luftkameras als Geoinformationssysteme. Flugzeugsoftware mit einer Messpräzision von 0,4 mm wurde verwendet, um Fotografien mit Bodenmessungen zu verknüpfen. Helikites sind günstig und sammeln mehr adäquate Daten als Flugzeuge. Ein Produkt der jüngsten Entwicklungen sind unbemannte Miniaturdronen. So wurde z.B. die Aeryon Scout Drone eingesetzt, um in zwölf Minuten ein 20 Hektar großes Grundstück mit einer Musterdichte von 2,54 cm zu kartographieren.

13.5. Mehrdimensionaler Würfel

Die elementare Baueinheit in einer mehrdimensionalen Datenbank sind Würfel (Würfel, Daten-würfel, mehrdimensionale Würfel, Hyperwürfel). Ein Würfel in einer mehrdimensionalen Datenbank besteht aus einer Menge von Dimensionen und Messgrößen.

Die Dimensionen eines Würfels sind die Kategorien, anhand welcher Daten kumuliert und analysiert werden sollen. Dimensionen ergeben sich aus relationalen Datenbanktabellen. Typische Dimensionen in multidimensionalen Datenbanken sind Zeit, Standort und Produkt. Dimensionen können aus einer Anzahl an Ebenen bestehen, welche die Daten weiter aufbereiten.

Bei Würfelmessgrößen handelt es sich um die quantitativen Daten, die es es zu analysieren gilt. Dimensionen leiten sich aus relationalen Datenbanktabellen ab. Typische Maßzahlen sind Verkäufe, Kosten, Preise, aber fast jede quantitative Zahl kann eine Messgröße eines multidimensionalen Würfels sein.

13.6. Daten in der OLAP-Datenbank speichern

Die Quellendaten aus der relationalen Datenbank können in einer dieser drei grundlegenden Methoden gespeichert werden:

Datenschranken enthalten kumulierte Daten, welche die Grundlage für multidimensionale Analysen bilden. Umfangreiche multidimensionale Anwendungen (zB informetry) enthalten Daten, welche sich auf verschiedene Kontexte beziehen, aber einige Dimensionen können geteilt werden. Es ist daher eher verständlich, einen einzelnen Datenwürfel für einen Kontext zu erstellen als einen Datenwürfel zu verwenden, welcher alle Kontexte umfasst. Dieser Datenwürfel besteht nur aus jenen dimensionalen Attributen, welche all ihre numerischen Attribute miteinander teilen. Das bedeutet, dass dimensionale Attribute die Grundlage eines Datenwürfels bilden. Wenn man Informationen aus mehreren Datenschranken herausbekommen möchte, können diese Attribute auf der Ebene einer oder mehrerer gemeinsamer Dimensionen verknüpft sein. Die Navigation zwischen Datenwürfeln nimmt keine Rücksicht auf den Benutzer.

Dimensionstabellen beschreiben spezifische Dimensionseigenschaften. Für jedes dimensionale Attribut gibt es maximal eine Dimensionstabelle. Anders als das ursprüngliche Sternmodell, kommen die Dimensionswürfel in diesem Modell ohne Dimensionstabellen aus. Auf schematischer Ebene besteht eine Dimensionstabelle aus Attributsnamen und auf der Beispielebene basiert es auf den Werten dieser Attribute.

In Informationen müssen kumulierte Daten oft auf Basis komplexer Kriterien analysiert werden, welche von den Eigenschaften der Dimensionen abhängen. Um diese Kriterien spezifizieren zu können, sollten die Informationen in mehreren Dimensionstabellen verwendet werden. Im Anschluss werden semantische Beziehungen zwischen Dimensionstabellen erstellt, welche auf gemeinsamen Werten einiger Attribute in den Dimensionstabellen basieren.

INFORMATIONS- UND KOMMUNIKATIONSTECHNOLOGIEN

1. Didaktische Mittel, Hilfsmittel, Multimedia

Die Informations- und Kommunikationstechnologien, IKT, umfassen alle Informationstechnologien, die für die Kommunikation und Informationsarbeit genutzt werden.

Komponenten didaktischer Techniken:

- Dies sind Didaktische Hilfsmittel (Techniken und technische Geräte zur Bereitstellung von visuellen, akustischen und audiovisuellen Informationen). Das Hauptmerkmal ist die absolute Universalität.
- Lehrmittel (alle entsprechend aufbereiteten Lehrpläne), hohe Spezifität

1.1. Didaktisches Hilfsmittel

1. Originelle Themen und reale Fakten

    - Naturprodukte im Ursprungszustand (Mineralien, Pflanzen, etc.), zubereitet (Zubereitungen, Füllungen, Stecklinge, etc.)
    - Produkte und Kreationen im Originalzustand (Instrumente, Kunstwerke, etc.), modifiziert (Sets und Mustermengen, Schneidemaschinen, etc.)
    - Phänomene und Prozesse, physikalische, chemische, biologische, soziale, etc...,
    - Klänge, Realsounds, Stimme und musikalische Ausdrücke.

2. Ansicht und Darstellung von Objekten und Fakten

- statische, funktionale, modulare, flache Modelle, etc...,
- direkt präsentierte Präsentationen (Bilder, Fotos, Diagramme usw.), die mit technischen Mitteln (statisch, dynamisch, interaktiv, virtuell, 3D usw.) präsentiert werden.
- Tonaufnahmen

3. Texthilfen gedruckt oder digital

- klassische, arbeitende, programmierte, interaktive Bücher,

- Arbeitsmaterialien, Wörterbücher, Tabellenkalkulationen, Aufgabensammlungen, Atlanten, etc...,
- zusätzliche und ergänzende Literatur- und Informationsquellen.

4. **Programme und Programme, die mit technischen Mitteln präsentiert (implementiert) werden.**

- Programme, Lehrfilme, Radio- und Fernsehprogramme, etc.
- Programme, Informationen, Nachhilfe, repetitive, etc.

5. **Spezielle Hilfsmittel**

- Versuchsaufbauten, Baukasten usw.

1.2. Lehrmittel

Es gibt zwei Aspekte bei der Beurteilung der Richtigkeit und Eignung des Lehrplans:

- **inhaltliche Sichtweise** - beurteilt, ob der in der Beihilfe enthaltene Lehrplan dem aktuellen Stand des festgestellten Problems entspricht und mit dem Ziel übereinstimmt, den gegebenen Schultyp zu unterrichten - eine Frage der Didaktik des gegebenen Faches.

- **formaler Gesichtspunkt** - löst die Frage, ob es möglich ist, den Schülern die mittels DT gegebene Hilfe (Schriftgröße, angemessene Größe und Form, etc.) - das Thema der didaktischen Technik - problemlos zu präsentieren.

Die Beihilfe dient der Information:

- Inhalte (bezogen auf die eingereichten Inhalte - Konzepte und Beziehungen zwischen ihnen, etc.)
- Sinnerfassend (zeigt, wie man mit den bereitgestellten Informationen umgeht).

Die Hilfe ist sehr spezifisch - sie respektiert das jeweilige Fach, die Art der Schule, das Jahr, das Thema und den spezifischen Lehrer.

Die Bedeutung und Funktion von Lehrmitteln und DT bedeutet im Unterricht:

- Helfen Sie mit, das Interesse der Schüler zu wecken,
- Förderung der Konzentration der Schüler,
- Fördern Sie die Konsistenz der Aufmerksamkeit,
- Motivations-Funktion,
- Führen Sie sowohl zu Lerninhalten als auch zu Arbeitsmethoden,

- Erweiterung der Informationsmöglichkeiten im Lehr- und Lernprozess,
- Erleichtert die Mediation von schwer verbal kommunizierten Personen,
- Ermöglichen Sie ein treueres Verständnis der Realität,
- Kombination der Theorie mit Praxis.

Modernisierung des Lehrprozesses:

Es ist nicht nur eine bloße Einführung moderner DT-Ressourcen in den Lehrprozess, sondern die nächste Abfolge von Phasen.

- Modernisierung der Inhalte des Lehrprozesses,
- Modernisierung der Lehrmethoden,
- Die Einbeziehung moderner DT-Mittel.

1.3. Multimedia

Sind der Bereich der Informations- und Kommunikationstechnologien, der durch die Zusammenführung audiovisueller technischer Geräte mit Computern oder anderen Geräten gekennzeichnet ist.

Ein Multimediasystem ist eine Sammlung technischer Ressourcen (wie z.B. ein Personalcomputer, eine Soundkarte, eine Videokarte oder eine Videokarte, eine Kamera, ein CD-ROM- oder DVD-Laufwerk, eine entsprechende Servicesoftware usw.), die für eine interaktive audiovisuelle Präsentation geeignet ist. Seit Anfang der 90er Jahre wird der Einsatz von Multimedia-Anwendungen oder Multimedia-Software eingesetzt, um Text-, Bild-, Ton-, Animations- oder Filmdaten zu kombinieren.

2. Didaktische Technik

Didaktische Geräte (DT): Geräte und technische Geräte zur Bereitstellung von visuellen, akustischen und audiovisuellen Informationen.

Voraussetzungen für den Einsatz von Lehrmitteln / DT /:

- Gründliche Analyse der Lektion: Bestimmung der Struktur und des Schwerpunkts der Lektion.
- Wir wählen Wissen aus, um die notwendigen Lernmittel zu klären.
- Wir werden die technische Form der Hilfe und ihre Einbeziehung in die Lehre aus ihrer Funktion im Prozess des Erlernens des gegebenen Wissens ableiten.
- Bewertung des Verhältnisses zwischen dem während des Unterrichts verwendeten Lehrmittel und den Materialien, die den Schülern für ihre eigene Hausvorbereitung zur Verfügung stehen (in Lehrbüchern, Notizbüchern usw.) Ist es ausreichend, das Lehrmittel während des Unterrichts zu verwenden, oder ist es angebracht, dass die Schüler für die Hausaufgabenvorbereitung zur Verfügung stehen?

2.1. Multimediale didaktische Mittel

Multimedia - Systeme für technische Elemente zur interaktiven audiovisuellen Präsentation; Ermöglichen Sie dem Benutzer die Arbeit mit verschiedenen Medientypen (Medien): Text, Bild, Ton, Computergrafik und Animation, Video und Interaktivität zwischen dem System und seinem Benutzer. D.h. Multimedia zeichnet sich durch die folgenden Merkmale aus:

Per Text ? Audio-visuell? Bild ? Animation ? Video ? Interaktivität ?

Die Interaktivität kann verschiedene Optionen beinhalten:

- der Nutzer wählt Inhalte aus, kombiniert deren Teile,
- der Benutzer beeinflusst die Geschwindigkeit, mit der das Multimediasystem Informationen präsentiert,
- der Benutzer tauscht Informationen mit dem System aus (z.B. durch Frage-Antwort).
- der Benutzer steuert den Prozess, durch den das System Informationen präsentiert,
- Der Benutzer fügt den Inhalt des Systems ein, vergleicht ihn, etc.

Multimediale didaktische Mittel

- Lernsoftware (z.B. interaktive Multimedia-DUMs), Multimedia-Präsentationen, Multimedia-Aufzeichnungen des Unterrichts - z.B. Videovorträge, didaktische Computerspiele,
- Learning Management System (LMS): Planung, Erstellung, Präsentation und Verwaltung von Inhalten sowie einer geeigneten Umgebung, mit Tools für LEKTOR und KUNDEN und deren gegenseitige Kommunikation (z.B. Moodle).

Verbreitung didaktischer Techniken (je nachdem, welche menschlichen Sinne wir bei der Anwendung von DT verwenden):

- Sehhilfen - sie beeinflussen das Sehen.
- Hörtechniken - sie beeinflussen das Hören,
- Audiovisuelle Techniken - sie betreffen Vision und Hören gleichzeitig,

Hypermedia und Hypertext didaktische Mittel

Hypertext - ein Text, der aus Wortblocks oder Symbolen besteht, die elektronisch durch Pfade (elektronische Linien) in einer offenen und noch unvollendeten Texturstruktur (Netzwerk) verbunden sind.

Hypermedia - Eine digitale Ressource, die aktive Links nicht nur zu Texten, sondern auch zu Tabellenkalkulationen, Animationen, Bildern, Audio und Video enthält.

Hypertext und hypermediale didaktische Ressourcen - Digitale Ressourcen, die Hypertext und hypermediale Elemente enthalten. Sie haben die Form eines Netzes, in dem die Hauptlinien des Textes miteinander verbunden sind.

Textzeilen und Medienelemente, die den Haupttext entwickeln, ergänzen, illustrieren. Das Lernobjekt geht in diesen Netzwerken auf eine einzigartige, individuelle, unvorhersehbare Weise vor.

3. Modellierung. Definition. Theoretische Modelle.

Der Begriff "Modell" wird in verschiedenen Kontexten verwendet. Dieser Begriff bedeutet:

- Mathematische und mentale Ansätze zur Lösung des Problems,
- Didaktische mittel

In der gesprochenen Sprache bedeutet der Begriff am häufigsten:

- Das ursprüngliche Muster von etwas Geschaffenem;
- Ein Prototyp, wie z.B. ein Automodell;
- Das ideale Muster, das wir in einer bestimmten Aktivität oder Funktion umsetzen wollen, wie zum Beispiel das Modell eines perfekten Lehrers;
- Die Struktur dessen, was in unserem Interesse liegt, wie z.B. das Modell der Schule;
- In der Kunst ist es eine Sache oder eine Person, welche gemalt, oder modelliert werden soll

Das ideale Modell ist eine Darstellung des untersuchten Phänomens, die hypothetische Erklärungen enthalten kann und helfen kann, die Richtigkeit der Hypothese zu überprüfen. Ideale Modelle, die mit einer bestimmten Theorie verbunden sind, werden als theoretische Modelle bezeichnet. Materialmodelle sind bestehende Objekte, deren Eigenschaften eine Rekonstruktion der Struktur oder des Wesens des zu untersuchenden Subjekts oder des Prozessablaufs ermöglichen.
Es handelt sich um vereinfachte und verzerrte Nachbildungen von Idealmodellen, die daher als "Modell von Modellen" erkannt werden können. Bevor das Materialmodell entsteht, muss es im Kopf des Wissenschaftlers als eine bestimmte Idee, d.h. als ideales Modell existieren. Daraus folgt, dass Modelle in ihrer ursprünglichen Form abstrakte Konzepte sind, die später konkretisiert werden können. Für wissenschaftliche und pädagogische Zwecke sind die Modelle am wichtigsten, die mit der Schaffung neuen Wissens verbunden sind.

Theoretische Modelle

Sie können die folgenden Funktionen ausführen:

- Informationen über die Theorie und ihre Verbindung mit relevanten experimentellen Daten an den Kunden weiterzugeben;
- Die Klienten mit Konzepten im Bereich der Theorie vertraut zu machen und die Fähigkeit zu entwickeln, Modelle zur Lösung vorgegebener Probleme einzuset-

zen;
- Verifizierung des Modells durch experimentelle Bestätigung oder Ablehnung der darauf basierenden Annahmen, um die Kunden mit dem Umfang und den Bedingungen seiner Nutzung vertraut zu machen;
- Verbalisieren Sie das Modell, was zur Formulierung der Annahmen der jeweiligen Theorie führt.

Modelle können in statische und dynamische Modelle unterteilt werden:
Statische Modelle - meist gefaltet und geschnitten, perfektes Bild und schnelleres Verständnis der Funktion.

Dynamische Modelle - sind Modelle, die Bewegung und Funktion nachahmen.

Simulatoren - sind Modelle von realen Objekten, Geräten usw., die für den Einsatz von realen Geräten vorbereitet und perfektioniert werden können.

Virtuelle Computermodelle - das sind Modelle, die einige Phänomene, Objekte, Objekte im dreidimensionalen oder zweidimensionalen Raum simulieren.

Funktion von Modellen, technischen Lehrmitteln

- Grundfunktionen: Information, Gestaltung, Instrumentalisierung.

- Didaktische Funktion: Motivations- und Stimulationsfunktion, Rationalisierung in Bezug auf Lehrer und Schüler, Verstärkung der Informationswiederholung, Systematisierung (Aufnahme von Informationen in das System des zuvor erworbenen Wissens), Steuerungs- und Kontrollfunktion.

- Ergonomische Funktion: (die Lehre von den Beziehungen zwischen Mensch und Arbeitsumfeld und den für das Arbeitsumfeld am besten geeigneten Arbeitsmitteln) Management: z.B. Reduzierung unnötiger Zeit für Lehrer und Schüler, volle Nutzung für das Lehrmanagement, Regulierung des eigenen Lernrhythmus entsprechend den Dispositionen und dem Zustand der Psyche.

Errungenschaften eines neuen, dynamischen Computermodells:

Die erste und wichtigste Annahme ist, dass das neu entwickelte Mikroskopmodell und die dazugehörigen visuellen Lehrmittel inhaltlich (verdienstvoll) korrekt sind.

4. Annahmen eines neuen dynamischen Computermodells - Visualisierung

Erscheinungsbild der Visualisierung

Um Objekte im Lernprozess zu betrachten, werden derzeit interaktivere 3D-Modelle eingesetzt, die unterschiedliche Darstellungsmöglichkeiten in unterschiedlichem Detaillierungsgrad bieten, im Gegensatz zur 2D-Visualisierung, die nicht klar genug ist und die Realität nicht vollständig widerspiegelt (manchmal kommt es vor, dass einige Prozesse, die im Hintergrund modelliert werden, versehentlich miteinander verschmelzen). Interaktive 3D-Modelle bringen uns der realen Realität näher, die wir in der realen Welt mit Schwierigkeiten sehen oder untersuchen können oder die nicht mehr existiert. Die gewonnenen Informationen können dann mit der "erlebten" Erfahrung in Verbindung gebracht werden, indem man sich besser an sie erinnert und sie bei Bedarf "umrüstet". In unseren Bildern spiegeln wir und unsere Erfahrung uns wider.

Im Bild schafft es ständig eine attraktive Kraft und ein Geheimnis auf der emotionalen Ebene, das den Menschen zu Interaktivität, sozialer Kommunikation und zur Schaffung von eigenem Wissen, Wissensstrukturen und kritischer Bewertung von Informationen anregt (Scruton, 2005).

Mittel der visuellen Technik

- <u>für nicht geschirmte Hilfsmittel</u>
  Mittel der DT Hilfsmittel (Informationsträger)Tafel Pädagogische Zeichnung Magnettafel Papiermaterialien

  Objekte auf Ferrit Flanellplatte Werkzeuge, die mit Flanell oder auf Klettverschluss geklebt sind Plexitplatte Pädagogische Zeichnung Flipchart Pädagogische Zeichnung Stand oder Haken auf der Platte Wandtafel, Bild oder Karte

- <u>für statische Projektion</u>
  DTA Hilfsmittel (Informationsträger) EpiprojektorEpiprojektorEin Werkzeug auf einem opaken PaddiaprojektorDia, DiabeltOverhead-ProjektorHilfsmittel auf einer transparenten Oberfläche oder einem großen Dia

- <u>für dynamische Projektion</u>
  Mittel der DT Hilfsmittel (Informationsträger) FilmprojektorStummfilm, Filmschleife

Konstruktivismus und Visualisierung

In den letzten Jahren → Umsetzung des Konstruktivismus in der technischen und multimedialen, humanitären und fremdsprachlichen Bildung.

Das konstruktivistische Konzept wird als ideale pädagogische Grundlage für die Visualisierung bezeichnet, d.h. für die Visualisierung der durch visuelle Rezeptoren wahrgenommenen Realität.

Konstruktivistische Pädagogik stellt den Kunden in den Mittelpunkt des Lernprozesses. Ebenso geht die mit der Anwendung des Prinzips der Klarheit verbundene Visualisierung von einem unabhängigen Kunden aus, der sein Lernen teilweise verwalten und organisieren kann.

Die traditionelle Rolle eines Lehrers ändert sich auf natürliche Weise wird zum konstruktivistischen Tutor, Moderator und Reiseleiter.

Die Kommunikation zwischen den Akteuren sollte sowohl durch die Schaffung einer angenehmen Atmosphäre des offenen Raums für den Meinungsaustausch als auch durch ein geeignetes Konzept von Gruppe / kooperativ oder kollaborativ / Arbeit gefördert und aktiviert werden.

Die Hauptmerkmale des konstruktivistisch konzipierten Unterrichts können als der Übergang von der transmissiven Lehre, der so genannten "Sie-Bildung", zur Selbstevaluation, d.h. Selbstorganisation, Selbstbestimmung, Selbstbildung, betrachtet werden.

Medienansätze

- Medienoptimismus - Die optimistische Rezeption der Medien, die die Medien als allgegenwärtigen Geschäftsmann betrachtet. Transhumanismus, Extropismus, Singularitarismus, Techno-Autopismus.

- Medienpessimismus - Kritische Gegner medienoptimistischer Richtungen, die auf die negativen Aspekte der Technologie- und Medienentwicklung hinweisen und die Fusion von Mensch und Technik (Medien) ablehnen.

- Mediakismus - zu viel Abhängigkeit von den Medien. Der Glaube daran, dass die Menschheit in der Lage sein wird, die technologischen Medien und alle damit verbundenen Probleme vollständig zu beherrschen und effektiv für ihr Wohlergehen zu nutzen.

5. Die Rolle des Bildes in der Kultur. Unterrichten mit einem Bild.

Bilder und Zeichnungen begleiten den Menschen seit dem Beginn der Zeit: Die ältesten Gemälde, die wir heute kennen, sind Gemälde in der Chauvet-Höhle in Frankreich (31.000 ± 1300 v. Chr.). Von diesem Moment an begleitet die Menschen ständig ein Bild, das im Laufe der Zeit verschiedene informative und ästhetische Funktionen hat. Die wichtige pädagogische Rolle der Visualisierung ist im Mittelalter zu erkennen. Typisches Beispiel für die damalige "Bilderziehung" kann die Bibel Pauperum (Bibel der Armen) sein.

Es war sehr häufig, das Bild mit dem Text zu verknüpfen, um eine bestimmte Geschichte besser zu veranschaulichen. Im Laufe eines Jahrhunderts erschienen in vielen Gemälden Symbole, Allegorien und Embleme, die den Text darstellten. Stattdessen wurde der Text selbst oft von Bildern begleitet, oft in Form von Initialen oder Illuminationen.

John Amos Comenius, der 1658 Orbis sensualium pictus schrieb, nahm auch die Tradition des Unterrichts mit Hilfe der Malerei auf. Es handelte sich um eine illustrierte "Enzyklopädie", die in 150 Kapitel mit 150 Holzschnitten unterteilt war. Die Welt auf den Bildern zeigte und benannte alle Dinge der geistigen und materiellen Welt, die für die Kinder notwendig waren.

Visualisierung

Bilder können nicht nur der Rolle der Illustration dienen, sondern manchmal ist es auch möglich, schwer vorstellbare Probleme zu erklären. Als Beispiel für einen solchen Ansatz war der Text von prof. Janusz Rachonia vom Polytechnikum Gdańsk, das beschreibt, wie er den Schülern die Resonanztheorie von Linus Pauling durch die Malerei von Salvador Dalí erklärt.

6. Medienkommunikation. Soziale Kommunikation: verbale, nonverbale Kommunikation.

Medienbildung

Die Beziehung zwischen Medienbildung und Medienkompetenz ist einfach eine Beziehung zwischen den Mitteln und dem Ziel. Medienkompetenz wird somit definiert als Medienkompetenz als Orientierungsbildung in den Massenmedien, deren Nutzung und gleichzeitig als kritische Einschätzung als bewusster pädagogischer Einfluss auf die Erreichung eines bestimmten Grades an Medienkompetenz oder einfach als Bildung für das Leben mit den Medien. Die Medienbildung wurde allmählich aufgebaut, und ihre Wurzeln liegen manchmal bei Comenius, manchmal bis ins antike Griechenland, aber die eigentliche Entwicklung kam nach dem Zweiten Weltkrieg. Medienkommunikation hat enorme Macht, sie schafft ihre Realität; Wird verwendet, um die Reichweite oder Selbstpräsentation zu erhöhen; Es ist gut, eine positive Beziehung zu den Medien aufrechtzuerhalten; Nachbarsektoren - öffentliche Realitäten und Medienrealitäten.

Arten von Kommunikationsmedien

- Primär - "Zeichensätze und Regeln für ihre Verwendung (Muttersprache)".

- Sekundär - Mittel zur Aufzeichnung und Übertragung von Nachrichten (Bilder, Schriften, Druck-, Übertragungs- und Rundfunktechnik, Computerkommunikationsnetze).

Das Wort Kommunikation kommt aus dem Lateinischen communicare, was bedeutet: "etwas miteinander teilen, etwas gemeinsam tun". Kommunikation ist eine Grundvoraussetzung für die Existenz jeder sozialen Beziehung. Es ist auch ein Mittel zur sozialen Integration des Einzelnen in die menschliche Gemeinschaft. Für eine perfekte Interaktion mit der Umwelt ist es notwendig zu lernen, auf den inneren Impuls zu hören - Ihre Gedanken bewusst zu beobachten, ein ständiges inneres Gespräch zu führen, Gefühle zu überwachen und zu verstehen.

Die Kommunikation kann unterteilt werden in:

- Intrapersönliche Kommunikation findet innerhalb eines Individuums statt und findet in Form eines internen Dialogs statt. Es ist ein "Selbstgespräch", eine Selbstreflexion des eigenen Verhaltens und der Kommunikation nach außen.

- Zwischenmenschliche Kommunikation findet zwischen zwei oder mehr Men-

schen statt, zwischen denen eine Beziehung besteht. Eine spezifische Art der zwischenmenschlichen Kommunikation ist die Gruppenkommunikation.

- Massenkommunikation ist gekennzeichnet durch einen einseitigen Informationsfluss von einem und mehreren Kommunikatoren (Ressourcen) zu vielen Kommunikatoren (Empfängern). Dies ist Kommunikation über Medien wie Radio, Fernsehen, Presse und Internet. In der Massenkommunikation gibt es kein direktes Feedback, daher ist es wichtig, die erhaltenen Informationen kritisch zu bewerten.

Vybíral (2009) definiert fünf grundlegende Kommunikationsfunktionen: informieren, instruieren, überzeugen, verhandeln, unterhalten. Soziale Kommunikation ist Voraussetzung und Voraussetzung für die Existenz einer menschlichen Gemeinschaft.

Kommunikationsfaktoren:

- Quelle der Kommunikation - Kommunikator,
- Ermittlung der Kommunikation / Empfänger / Empfänger / Empfänger
- Kommunikant,
- Kommunikation / Kommuniqué / Kommuniqué /,
- Kommunikationsraum - Kanal.

7. Medienerziehung als Verteidigung gegen die negativen Auswirkungen der Medienkommunikation.

Die Beziehung zwischen Medienlehre und Medienkompetenz ist einfach eine Beziehung zwischen den Mitteln und dem Ziel. Medienkompetenz wird somit definiert als Medienkompetenz als Orientierungsbildung in den Massenmedien, deren Nutzung und gleichzeitig als kritische Einschätzung als bewusster pädagogischer Einfluss auf die Erreichung eines bestimmten Grades an Medienkompetenz oder einfach als Bildung für das Leben mit den Medien. Die Medienbildung wurde allmählich aufgebaut, und ihre Wurzeln liegen manchmal bei Comenius, manchmal bis ins antike Griechenland, aber die eigentliche Entwicklung kam nach dem Zweiten Weltkrieg.

Gründe für die Medienbildung

- Viele Informationen, Wissen, Meinungen;
- Die sich ändernden Bedingungen des menschlichen Lebens;
- Entwicklung der Medienkompetenz im Rahmen der Allgemeinbildung;

Medienkompetenz = eine Reihe von Kompetenzen, die dem Nutzer helfen, Informationen zu suchen, zu analysieren, zu bewerten und weiterzugeben;

Der Inhalt und die Art der Umsetzung sind unterschiedliche Bildungssystem.

Konzeptionelle Fragen:

- Was sollte der Inhalt der Medienbildung sein?
- Was ist die Mission der Medienbildung?
- Wie wird Medienbildung realisiert?

Konzept der Medienbildung

Zwei Grundkomponenten:

- Wissen - typisch für Kanada und Skandinavien - basiert auf einem kritischen Zweig von Medientheorien, der Frankfurt School und der britischen Kulturwissenschaft.

- Kompetenz - typisch für die USA, setzt voraus, dass die Schülerinnen und Schüler die notwendigen Kenntnisse und Fähigkeiten erwerben, indem sie ausprobieren, wie die Medien funktionieren.

Negative Auswirkungen der Medienkommunikation

Radio

Ich glaube, dass die Rolle des Radios als Mittel zur Beeinflussung der Ansichten und des Wertesystems von Kindern und Jugendlichen derzeit die kleinste aller Informationstechnologien ist. Die meisten Jugendlichen nutzen das Radio nur als Klanghintergrund. Das Einzige, was den Hörer beeinflussen kann, sind die notwendigen Anzeigen, die oft durch die melodiösen Melodien und das Gedächtnis des Publikums sehr gedämpft werden, und einige Slogans werden ständig in Erinnerung behalten.

Fernseher

Eines der wichtigsten Medien über die Psyche des Kindes ist das Fernsehen. Wenn ein Kind 24 McLUHAN TV schaut, Herbert für einen Actionfilm mit schlechtem Inhalt, hält er alles, was dazugehört, für selbstverständlich und will es auch ausprobieren, ohne auch nur an die Folgen zu denken.

DVD-Player

Kinder verbringen ihre Zeit damit, DVDs anzusehen, anstatt mit Freunden auszugehen, aber dieses Hobby kann auch dazu führen, dass DVDs kopiert werden, die aus urheberrechtlicher Sicht illegal sind. Kinder, die sich einen Film auf DVD ansehen, entwickeln ihre rhetorischen Fähigkeiten nicht, sie verbinden keine Fantasie. Das kann zu kleinem Wortschatz, schlechter Sprache und Fettleibigkeit führen.

Handy

Mit einem Mobiltelefon besteht die Gefahr, dass Schüler ihr Handy zum Betrügen benutzen. Es führt zu Unachtsamkeit während einer Stunde, zu Konzentrationsschwäche. Wird oft verwendet, um Spiele zu spielen, SMS zu schreiben, während des Unterrichts.

Ein Beispiel für negative Auswirkungen

- Die tägliche Kommunikation erfolgt über E-Mail,
- Wir verwalten die Finanzen mit E-Banking,
- Wir lesen Nachrichten meist im Internet,
- Der Einfluss der Medien auf die Psyche des Einzelnen - insbesondere bei Kindern und Jugendlichen,
- Soziale Netzwerke - Kinder,
- Der effektivste Weg, die negativen Auswirkungen zu lindern, ist die Familie.

8. Fotos, seine Geschichte. Digitale Gegenwart und Zukunft.

Die Geschichte der Fotografie

Sogar die alten Griechen haben herausgefunden, dass, wenn das Licht durch ein kleines Loch im dunklen Raum geht, es das Bild dessen zeigt, was da draußen ist. Im 9. Jahrhundert v. Chr. wurde diese Erfindung von Arabern in der Astronomie verwendet, um den Standort der Sonne oder der Sonnenfinsternisse zu bestimmen. Leonardo da Vinci, ein italienischer Gelehrter des 15. Jahrhunderts, beschrieb das Phänomen ausführlich als "Camera Obscura" oder Dunkelkammer.

Im 16. Jahrhundert benutzte der Italiener Giambattista Della Porta ein Objektiv anstelle eines kleinen Lochs, um ein schärferes Bild zu erhalten. Seitdem ist Camera Obscura in allen Größen erschienen, von den kleinsten Größen bis hin zu den großen auf den Aussichtstürmen und Leuchttürmen.

Im 16. und 17. Jahrhundert kamen Chemiker zu dem Schluss, dass einige Substanzen, wenn sie im freien Raum gelassen wurden, ihre Farbe änderten.

1725 - Johann Heinrich Schulz entdeckte, dass Silbersalze lichtempfindlich sind. Es dauerte jedoch fast hundert Jahre, bis die Kombination dieser beiden separaten Entdeckungen das erste Bild auf Papier schuf.

Mit der Erfindung der Fotografie entstand eine neue Art von Bild, bei der die Technik und Mechanik, die die manuelle Arbeit bei der Erstellung einer Wirklichkeitsaufzeichnung übernahm, eine wichtige Rolle bei ihrer Erfindung spielte. Die neue Technologie basierte nicht mehr auf der Handarbeit, sondern auf dem photochemischen Prozess.
Digitales Foto

Das Aufkommen der Digitalkameratechnologie ist mit der Fernsehbildaufnahmetechnik verbunden. 1951 nahm der Tonbandgerät (VTR) erstmals das Bild der Fernsehkamera durch Umwandlung in elektrische Impulse auf und legte es in den Labors von Bing Crosby (einem von Crosby gesponserten Forschungsteam unter der Leitung von John Mullin) auf ein Magnetband ab. 1956 verbesserte sich die FTE-Technologie durch die Entdeckung von Charles P. Ginsburg (Ampex Corp.) und kam in der Fernsehindustrie zum Einsatz. TV- und Videokameras verwenden eine sehr ähnliche Technologie wie CCD-Scanner für Digitalkameras.

Mitte der 70er Jahre entdeckte Kodak mehrere Sensoren, die statische Bilder erfassen konnten, die auch nach dem Prinzip der Lichtübertragung arbeiteten. 1986 entdeckten die Kodak-Forscher den weltweit ersten Megapixel-(MPix)-Sensor, der 1,4 Millionen Pixel

aufnehmen kann (d.h. ein Foto von 10x15 cm in Fotoqualität aufnehmen). 1990 entwickelte Kodak das Photo-CD-System - den weltweit ersten Standard für die Farbdefinition im digitalen Umfeld von Computern (und Peripheriegeräten - Druckern). Vor allem in den Jahren 2000 und 2001 sind digitale Fotos auf der technologischen Seite entstanden. Von 0,3 MPix-Sensoren bis hin zu handelsüblichen professionellen CMOS-Sensoren mit 22 MPix Auflösung. Von Kunststofflinsen bis hin zu High-End-Ultraviolettlinsen.

Wichtige Persönlichkeiten der Fotografie

- Joseph Niecephore Niepce
- Luis Jacques Mandel Daguerre
- F.S. Bogenschütze

9. Film, seine Geschichte, Entwicklung und Zukunft. Filmtechnik.

Der Film ist ein zusammengesetztes Zeichen für die Kinematographie und all ihre Aspekte - künstlerisch, technisch, kommerziell und sozial.

Art der Informationsweitergabe, Quelle der Unterhaltung, Kommunikationsmittel.

Während der Betrachtung des Films sieht das menschliche Auge eine Folge von aufeinanderfolgenden Bildern. Die schnelle Rotation der projizierten Bilder in sehr kurzer Zeit mit unserem Augennerv und damit das Gehirn erscheint als ununterbrochene Aktivität (das Bild ist nicht zerrissen und wird vom Gehirn als kontinuierliche Bewegung wahrgenommen).

Historie

- 1893 - Kinetoskop - William Kennedy Laurie Dickson.
- Viewer für einen Zuschauer.
- Der Film konnte nur durch ein kleines Guckloch am Gehäuse des Gerätes betrachtet werden.

Die Entwicklung des Films als technische Erfindung als eine Art Kunst-, Kultur-, Sozial- und Wirtschaftsphänomen erfolgte Ende des 19. Jahrhunderts. Die Geschichte des Films als eigenständige künstlerische Form beginnt 1895, als ein Kameramann der Brüder Lumičre (28. Dezember 1895 in Paris) in Paris mit großem Erfolg präsentiert wurde. Der Film war 35 mm breit und 16 Bilder pro Sekunde (fps). Die Gebrüder Lumiers beschäftigten sich mit Dokumentar- und Reisevideos. Der erste Kurzspielfilm gilt als ihre Upper Spider. Komödie mit Gärtner und Bewässerungsschlauch.

Der Film entstand als technisches Wunder, als jüngstes, synthetischstes und audiovisuellstes Kunstwerk, dessen Hauptmerkmal die Ikonizität ist. Der Erfinder des Films ist T.A. Edison, aber die Motive des Films finden sich in sogenannten animierten Fotos. Es gibt zwei grundlegende Paradigmen: die passive Erfassung der Realität (Dokumentarlinie - Lumiere) und die Science-Fiction, die Tricklinie (Mélies). Georges Méličs - 1897 das erste Filmstudio in Europa. Autor von Filmtricks.

In den 20er Jahren des 20. Jahrhunderts gab es einen Wandel, der die Form des Films völlig veränderte. In Kalifornien gab es eine Zeit, in der man Filmstudios gründen konnte. Die Qualität der Filme spielte eine immer wichtigere Rolle im Wettbewerbskampf, und es war notwendig, das Publikum mit etwas zu gewinnen, das sich von der durchschnittlichen Produktion unterschied.

Verschiedene Filmgrößen und Filmschnitte ließen die Filmspannung wachsen und führten zu Erzählungen auf mehreren Ebenen, die zum Beispiel im Theater schwer zu realisieren waren. Vertiefung und Verfeinerung des Filmausdrucks im Bereich der Bearbeitung und Nutzung der Kamera. Durch die Verwendung mehrerer Objektive wurde die Bandbreite der Aufnahmen von großen Einheiten bis hin zu Details voll ausgeschöpft, und die Kamera wurde auch vom Platz genommen und zum Fahren verwendet. In dieser Zeit wurde der Film vollständig vom Theater getrennt, er fand seine visuell-rhythmische Ausdrucksweise, aber auch eine neue filmische Handlungsweise, indem er Szenarien schrieb,

Stummfilm

Charlie Chaplin - einer der berühmtesten und bestbezahlten Künstler. Jedes Mal spielte er sozial deklassierte Charaktere, die gegen die schönen und erfolgreichen Menschen kämpfen.

Joseph "Buster" Keaton, schuf eine neue, völlig eigenständige Art der Filmkomödie. Der Charakter seines Filmcharakters war ein starker, steinerner Ausdruck seines Gesichts, das zu seiner "Unternehmensmarke" wurde und für das er den Spitznamen "The Great Stone Face" erhielt.

Tonfilm

Anfang der 90er Jahre entwarfen Edison und Dickson einen Prototyp eines Tonfilms, ein System namens Kinetofonograph oder Kinetofon - ein Vorfahre des Kineskops, das nichtsynchronen Klang erzeugt. Der erste bekannte (und auch der einzige überlebende) Film mit Live-Aufzeichnung war der Kinetophon-Test - Dicksons 17. Gesamt-Soundtrack.
The Jazz Singer (1927), gegründet von Warner Bros., begann die Ära des Soundtracks. In den nächsten fünf Jahren haben sich alle Technologien, Instrumente und das Gesicht des Films grundlegend verändert. Der Sound machte das Publikum mehr für den Film interessiert. Viele Schauspieler, die kein Englisch sprachen, verloren jedoch ihre Arbeit, ebenso wie Pianisten, die in stillen Kinos Soundtracks drehten.

Der sowjetische Soundtrack erschien etwas später. Diese Verzögerung wurde durch die Produktion von eigenem Filmmaterial verursacht, das es den Sowjets nicht erlaubte, Gebühren für Fremdmaterial zu zahlen. Noch 1930 wurden Stummfilme wie der Ball gedreht. Der erste englische Soundtrack wurde 1929 von Hitchcock aufgenommen.
Die 1930er Jahre waren eine große Hollywood-Ära, als viele neue Genres auftauchten und allmählich zu einer "Traumfabrik" wurden, deren spannende oder lustige Filme den Menschen eine kurzfristige Flucht aus der deprimierenden Realität ihres Alltags erlaubten. Fast alle Genres reproduzierten den Mythos vom Land der grenzenlosen Möglichkeiten und schworen in unendlichen Variationen den "amerikanischen Traum" einer großen Karriere "von der Spülmaschine zum Millionär".

Filmproduktion in der Tschechoslowakei

Ende 1941 wurden alle tschechischen Produktionsfirmen zerstört und nur noch zwei Produktionsfirmen - Lucernafilm und Nationalfilm - zurückgelassen. Alle Verleihungen wurden in das Kosmos-Monopol vereinigt, das neben Lucernafilm und Nationalfilm das einzige Recht hatte, tschechische Filme zu vertreiben.

10. Auditorische Medien, ihre Geschichte, Gegenwart und Zukunft

Hörhilfen

Laut Janíková (2005) spielen Hörgeräte eine wichtige Rolle bei der "Vermittlung und Übung des phonologischen Aspekts der lexikalischen Einheit". (Janíková 2005, S. 129) Sie helfen auch den Schülern zu Beginn des Fremdsprachenlernens und wenn sie mit dem neuen Vokabular vertraut sind. Bei der Verwendung von authentischem Material kann der Schüler die Aussprache von Muttersprachlern mit Hilfe von Hörhilfen hören. Sie können Aufnahmen auf einem Tonbandgerät oder einer CD, Nachrichten, Radio- oder Fernsehwerbung und mehr verwenden. Für viele Schüler bietet das Hören eines authentischen Aufzeichnungsmediums, wie z.B. eines Muttersprachlers, akustische Medien.
Mittel der Audiotechnik

Mittel der DT Hilfsmittel (Informationsträger) Grammophonophonaufzeichnung, Disketten-recorder, Tapecassettenrecorder, Tonbandgerät, CD-Player, D-Player, CD-Disc, Wireless-Set, Radiosendung, Audiobücher, MP3

Audiovisuelle Geräte

Mittel der DT Hilfsmittel (Informationsträger) Filmprojektor, Tonfilm, TV, TV-Übertragung, Videorecorder, Videoband, Multimedia-Computer, CD-ROMs mit Multimedia-Programmen, Datenprojektor, Daten in Computern und anderen Quellen, DVD-Player, DVD

Auditive Medien

Auditiv = Hören; Medien = Medium, dazwischen, in der Mitte. Die Medien sind ein Vermittler, sie vermitteln etwas.

Definition - es ist schwierig, den Begriff zu definieren; Es gibt viele Definitionen. Die Rolle der Medien: Kommunikation, Informationstransport. Der Begriff "Medium" bezieht sich auf technische Mittel und das System der sozialen Institutionen, die der Kommunikation dienen. Eine weitere mögliche Definition dieses Begriffs ist, dass "die Medien soziale Institutionen sind, die eine große Rolle bei der Sicherstellung der Kommunikation im öffentlichen Raum spielen und so zur Entwicklung, Etablierung und Transformation von Kultur, d.h. gemeinsamen Werten, Werten und Interpretationen der Welt, beitragen" (JIRÁK, J., KÖPPLOVÁ, B., 2003. 208, S.52).

Welche audiovisuellen Medien gibt es?

Dieser Begriff hat zwei Bedeutungen. Wir können Medien mit audiovisuellen Inhalten kennzeichnen - also (z.B. DVDs, Videokassetten), oder es ist ein Kommunikationsmedium, das auf diese Weise wirkt. 4 Milan Šmíd erklärt, dass es sich um einen Fachbegriff in der Abteilung der Medien handelt, der in Frankreich verwendet wird, wo zu den audiovisuellen Medien auch das Radio gehört, was etwas problematisch sein kann, da das Radio nur Hörinformationen sendet.

Andererseits unterteilt die angelsächsische Literatur die Medien in gedruckte und elektronische Medien. Zu den elektronischen Medien gehören Rundfunk-, Fernseh-, audiovisuelle und akustische Aufzeichnungen. Die amerikanische Literatur in dieser Abteilung beinhaltet unter den elektronischen Medien einen Film, der in der Vergangenheit nur wenige Gemeinsamkeiten mit der Elektronik hatte. Derzeit ist sie jedoch bereits durch enge Verbindungen mit den elektronischen Medien verbunden.

Gegenwart

Auditive Medien - immer noch verwendet, oft in Verbindung mit visuellen audiovisuellen Medien.
Spieler - Telefone, Tablets, Computer, Fernsehen, mp3 / 4, Spieler, Radio, Radio, Radio, Diktiergeräte, Mann :-)

Hörbuch = Tonaufnahme des gesprochenen Wortes; Hat bereits in der Geschichte begonnen, boomt jetzt dank Smartphones, Internet-Downloads.

Zukunft

Erwarten wir eine Zukunft für Hörmedien? Wird ein Fernseher oder ein Radio Teil unseres Hauses sein, wie es bisher ist? Werden die Fernseher in unseren Häusern durch Projektoren ersetzt?

Es gibt ein neues Konzept: Projektoren, die wenig Platz für die Darstellung benötigen - ein riesiges Bild in 25cm Entfernung, aufwändige Installation, Verbindung zur Set-Top-Box, Computer oder Spielkonsole mit Wireless-System; Projektoren für Mobiltelefone und Tablets / Lenovo,.... /. Die Senkung des Preises für Projektoren ermöglicht den Austausch von Fernsehern.

11. TV-Welt in der Geschichte. Analog und digital.

Geschichte des Fernsehens in der Tschechoslowakei

Pioniere
František Pilát - der spätere technische Direktor des Filmstudios Barrandov, baute selbst einen Fernseher. Pilatus war der erste in der Tschechoslowakei, der die experimentelle Sendung "thirty-line" von Baird erhielt, die in den frühen 1930er Jahren (1929-1935) von Großbritannien bis zu einer Mittelwelle von 261,5 Metern ausgestrahlt wurde.

Der aktivste Vorkriegspionier des Fernsehens ist dr. Jaroslav Šafránek, außerordentlicher Professor für Experimentalphysik an der Karlsuniversität in Prag. 1935 baute Šafránek seine eigenen funktionierenden Fernsehgeräte, mit denen er später in die Republik reiste und sie öffentlich präsentierte. Das Ministerium für Post und Telegrafie weigerte sich, Šafránek zu ermächtigen, das Fernsehen in der Luft zu übertragen. Šafráneks equipment konnte nur in den Labors und Hörsälen arbeiten. Während Šafránek, Funkamateure und ihre Interessengemeinschaft, die Tschechoslowakische Rundfunkgesellschaft, die Erlaubnis zur experimentellen Ausstrahlung eines mechanischen Niederzeilenfernsehers (30 Zeilen) beantragte, der hauptsächlich für Funkamateure bestimmt ist, wollte das Ministerium für Post und Telegraf, das seit 1934 die Entwicklungen im Ausland genau beobachtete, Frequenzen für die Fernsehübertragung für einige weiter entwickelte Projekte bereitstellen. Es wurde nach dem Prinzip geleitet - zu warten, ausländische Fakten zu studieren und dann zu entscheiden.

1939 wurde die Fernsehforschung auf dem Gebiet der ehemaligen Tschechoslowakei abgeschlossen. (Bedrohungen für die Republik, München und die Nazi-Okkupation). Zu dieser Zeit arbeitete Šafránek an einem weiterentwickelten 240-Linien-Bildzerlegungsgerät.

Am 17. November schlossen die Deutschen tschechische Universitäten ab.

Šafráneks Fernsehversuche endeten. Er verlor seine Stelle an der Universität, und sein Deutsches Physikalisches Institut wurde von den deutschen Behörden geschlossen. Šafránek soll es gelungen sein, einige Geräte in die Pardubicer Telegrafenfabrik zu bringen, wo er während des Krieges blieb.

Noch vor Kriegsende, im April 1945, verließen die deutschen Spitzenexperten die Fernseh A. G. Sie ziehen nach Österreich. Im Mai wird die Fabrik von tschechischen Behörden gemietet, die Fabrik in Smržovka Fernseh A. G. wurde sofort in Televid umbenannt. Anfang Juni wird Televid vom tschechoslowakischen Verteidigungsministerium unter seiner Leitung übernommen.

Im Juli 1945 wurde die Sicherheit des Unternehmens jedoch von der sowjetischen Militärverwaltung übernommen, für die die Fabrik Teil der Kriegsbeute war, und mehrere sowjetische Experten kamen vom Leningrader Fernsehinstitut. Zu diesem Zeitpunkt kommt Associate Professor Šafránek oft aus Prag nach Smrzovka. Aber nach den Erinnerungen griffen Zeugen nicht in die technische Entwicklung ein, weil sein mechanisches 240-Linien-System veraltet war und Televid an einem elektronischen, später "europäischen" Standard von 625 Linien arbeitete. Der Name Šafránek ist jedoch wieder einmal in die Geschichte unseres Fernsehens eingegangen. Er organisierte das Praktikum einer Gruppe von 25 Experten, die - nach Absprache der tschechischen Behörden mit der sowjetischen Militärverwaltung - im Oktober 1945 zu Smržovka kam. Bevor die Auszubildenden aktiv arbeiten konnten, beschloss die Sowjetische Partei, Televid als Kriegsbeute zu bewegen.

Jaroslav Šafránek wird dem Primat der Popularisierung des Fernsehens in der Tschechoslowakei zugeschrieben. Er veröffentlichte das Buch Televise, in dem er sich mit den technischen Grundlagen der Bildübertragung aus der Ferne vertraut machte. Eine aktuelle Version mit dem Titel "Televise - The Physical and Technical Foundations of the Television" von Šafránek wurde nach dem Krieg veröffentlicht. Šafránek versuchte, die einfache Technologie der Übertragung des bewegten Bildes vom komplexen Fernsehübertragungsprozess für das Fernsehen als Massenmedium zu unterscheiden, wo er das Wort "Brüllen" prägte, das seiner Meinung nach "das Wesen des Fernsehens richtig beschreibt".

Am 23. März 1948 wurden Journalisten nach Tanvald eingeladen, wo sie von VTÚ-General Josef Trejbal und dem technischen Stellvertreter des Tschechischen Rundfunks - Kazimír Stahl - begrüßt wurden. Als Teil der gezeigten Technik war das Design selbst, auch unter Verwendung einiger Trophäenkomponenten, ein Fernseher mit einem 16x21 cm großen eigenen Produktionsbildschirm.

Auf der MEVRO in Prag gab es zwei Fernsehkameras und das Signal wurde per Kabel an die Empfänger übertragen. Einen Monat nach dem Ende der MEVRO-Ausstellung, 4. Juli 1948, bei Transfers aus dem XI. Sokol-Organisationstreffen, drei Kameras arbeiteten am Strahov-Stadion, und das Signal wurde per Luftweg vom Mast von Petřín für 25 Empfänger in verschiedenen Institutionen und an öffentlichen Orten übertragen (z.B. Messegelände, ČSS Radio, Red Law Paper Redaction, Empfang wurde auch außerhalb von Prag in Südböhmen und im Riesengebirge überprüft).

Von 1949 bis 1952 gab es in der Tschechoslowakei kein Fernsehen mehr.

Die Fernsehausrüstung des Militärtechnischen Instituts VTU, darunter zwei Kameraketten und zehn Fernseher, wurde auf den Tschechoslowakischen Rundfunk übertragen, der Anfang 1949 vom Institut für Funktechnik ÚRT gegründet wurde. Obwohl er weiterhin die Aufgabe der Vorbereitung von Fernsehsendungen während der ersten fünf Jah-

re, d.h. bis Ende 1953, wahrnimmt, der Kalte Krieg, der 1950 durch den Konflikt in Korea verschärft wurde, dazu führte, dass das Büro ÚRT kein anderes Büro hatte, mit dem es zusammenarbeiten konnte, hat sich die technische Forschung ausschließlich auf militärische Bedürfnisse konzentriert. Laut Ing. František Křížka 1951 organisierte das Büro des Tschechischen Rundfunks in seinem Gebäude in Vokovice mehrere experimentelle Übertragungen und verlieh die Empfänger an Partei- und Regierungsbeamte, um das Fernsehen erfolglos zu fördern. Der Umsatz erfolgt 1952.

Am 8. April 1952 erließ die Regierung eine Verordnung, nach der das Ministerium für Kommunikation und Soziales "technische Radio- und Fernsehgeräte bauen und betreiben muss".

Der erste "Programmdirektor und Leiter der Fernsehwissenschaft", der im tschechoslowakischen Radio Karel Kohout angesiedelt war, wurde am 1. Februar 1953 (drei Monate vor dem geplanten Beginn der Sendung) ernannt. Karel Kohout kam aus dem Studio von Barrandov und begann mit seiner legendären Sekretärin Maria Kořenová, um eine Übertragung aus einem temporären Büro am Wenzelsplatz zu organisieren.

Seit einigen Jahren ist die Sendung auf den Sender Petřín in Prag beschränkt, der die mittelböhmische Region bis zu den Ausläufern des Isergebirges und des Riesengebirges erreicht. Ende 1953 waren rund 2000 Fernseher in Betrieb, von denen tausend Leningrader Marken aus der DDR importiert wurden, wo sie dann in sowjetischer Lizenz produziert wurden. Doch bereits 1953 brachte Tesla den Tesla 4001A auf den Markt. Sie verkauften für 4.000 CZK (damals war es ein fast halbjährliches Durchschnittsgehalt). Anfang Januar 1955, als die so genannte Konzessionsgebühr anfing, wurde die Statistik von 3833 Konzessionären gemeldet. In den späten 1950er Jahren wurde der Fernseher zu einem knappen Produkt auf dem Markt.

Am 11. Februar 1955 fand die erste direkte TV-Übertragung eines Sportspiels in der Geschichte des tschechoslowakischen Fernsehens statt.

Am 17. April 1955 fand die erste Direktübertragung der Oper aus dem Nationaltheater statt.

Seit Oktober 1955 wird es 6 mal pro Woche (nicht montags) ausgestrahlt; seit dem 29. Dezember 1958 wird es täglich, sieben Tage die Woche, bundesweit ausgestrahlt.

Am Silvesterabend des 31. Dezember 1955 nahm der zweite tschechoslowakische Fernsehsender Ostrava-Hošťálkovice den Betrieb auf.

Das Fernsehstudio in Brünn wurde erst 1961 gegründet; am 25. Februar 1962 begann die Fernsehübertragung in Košice.

Diese Grundstruktur der fünf großen Fernsehstudios widerstand in der Tat bis zur Auf-

lösung der Tschechoslowakei 1993. Um die Wende der 1950er und 1960er Jahre wurde ein Netz von Sendern und Konvertern aufgebaut, so dass 1961 das Fernsehsignal alle regionalen Zentren und den größten Teil der Tschechoslowakei abdeckte, so dass es kurz darauf (1962) mehr als eine Million Fernsehbesitzer gab.

Am 1. Januar 1993, nach der Auflösung der Föderation, wurde das Tschechische Fernsehen gegründet. In den 90er Jahren wurde das erste private Fernsehunternehmen gegründet (NOVA und andere). Im Jahr 2000 bereitete sich die Tschechische Republik auf den Übergang zur digitalen Fernsehübertragung auf der digitalen terrestrischen (ehemals terrestrischen) DVB-T-Plattform vor, die den analogen terrestrischen Rundfunk ersetzen sollte. Technisch ist die Tschechische Republik sehr gut vorbereitet - die experimentellen digitalen Fernsehsendungen waren in den drei größten Städten bereits erfolgreich.

Der Unterschied zwischen analogen und digitalen Übertragungen kann mit dem Senden von Geld durch einen Bus oder per Banküberweisung verglichen werden. Wenn wir das Geld an einen Schuhmacher schicken, erhält der Empfänger unser Geld physisch so, wie wir es geschickt haben, vorausgesetzt, der Coach erklärt nicht, dass der Kurier nirgendwo verloren geht, etc. Während wir sie per Banküberweisung versenden, erhält der Empfänger physisch anderes Geld, aber auf jeden Fall zum gleichen Wert wie beim Versand.

Die digitale Übertragung von Informationen ist die Übertragung eines Wertes in Form einer Zahl. (In diesem Fall die sogenannte Binärzahl).

Eine analoge Übertragung ist eine veraltete Form der Verbreitung von Fernseh- und Radiosignalen. Sowohl das Bild als auch der Ton werden durch elektromagnetische Wellen übertragen. Durch die Modulation dieses kontinuierlichen (analogen) Signals werden Farb- und Toninformationen erzeugt. Jede TV- oder Radiofrequenz trägt also ein Stationssignal. Der analoge Rundfunk wird derzeit durch den digitalen Rundfunk ersetzt. In der Tschechischen Republik wurde der analoge Rundfunk Ende 2011 eingestellt, als es zu einer analogen Dunkelheit kam.

Digital Broadcasting (DVB = Digital Video Broadcasting) ermöglicht es dem Multiplex, mehrere TV-Programme auf einer Frequenz zu übertragen. Dies macht es einfacher, die Bandbreitenauslastung zu nutzen.

12. Multimedia im Spiegel der Zeit. Medienphilosophie.

Welche Medien gibt es? Medien werden oft als Kommunikationsmittel, Medien, meist als technisches Mittel zur Kommunikation zwischen dem Kommunikator und dem Empfänger bezeichnet. Darüber hinaus sind sie Massenmedien, sogenannte Promotoren. Andere Medien als die Massenmedien spielen eine wichtige Rolle bei der Promotion. Und als Beispiele können wir Messen, Ausstellungen, Vitrinen, Cover, aber auch Vorträge, Exkursionen präsentieren.

Zeitungen, Zeitschriften, Filme, Radio- und Fernsehsendungen, das Internet (Nachrichten, Bildung, Unterhaltungsportale, Social Networking, Blogs, gesprochenes Wort, Filme, Musik) - Kommunikationsmittel, die einer potenziell großen Anzahl von Nutzern in regelmäßigen Abständen zur Verfügung stehen.

Durch sie können wir gezielt die öffentliche Meinung beeinflussen, Manipulationen, bewusste Auswahl und Schrumpfung von Nachrichten, Vermischung von Nachrichten und Bewertung von Kommentaren können auftreten, Übertreibung (positiv, Nachrichten klingen positiver, negative Nachrichten klingen negativer).

Die Kraft der Bearbeitung (nur ein Teil des Satzes bleibt übrig), die Manipulation mit dem Bild (die Reihenfolge der Bilder kann zu Verwechslungen von Ursache und Wirkung führen). Sprachausgabe.
Medienfunktionen

Informationsfunktionen

Es sind vor allem Nachrichten und Journalismus, in denen Nachrichten, Kommentare und Berichte verwendet werden.

Funktionsfuktion

Die Massenmedien bringen Musik, Kunst, Sport und sogar die heutigen Phänomene - Fernsehsendungen, Reality-Shows, Kochen.

KOMERČNÍ FUNKCE stellt den Medien Mittel für den Rundfunk zur Verfügung und garantiert den Gewinn, der hauptsächlich durch Werbung, aber auch durch Teleshopping und Sponsoring realisiert wird.

Politische und soziale Reichweite

Von 1900 bis 1925 entstanden die journalistische Industrie, die ersten Erfahrungsberichte und der Fotojournalismus.

1918 - Gründung der tschechoslowakischen Pressestelle "CTK"; Politische Tagebücher, freie Tagebücher, Tablare. Obwohl das System der Ersten Republik heute für viele ein Modell des demokratischen Systems ist, war das Funktionieren der Zeitschriftenpresse zu diesem Zeitpunkt viel eingeschränkter als heute.

Rolle des Radios - Unterhaltung, Nachrichten, Nachrichten

1925 begannen die Nachrichten im Radio eine neue und aktivere Rolle zu spielen, indem sie nicht nur Nachrichten berichteten, sondern versuchten, die Ereignisse näher zu bringen. Erste Versuche der Berichterstattung. Der erste Sportbericht.

Radio und Bildung

Nach und nach wurden auch Radiokurse für Fremdsprachen wie Französisch in das Programmangebot aufgenommen. Vorträge, Interviews, Diskussionen, Bands, Hörspiele, professionelles Fernsehen.

1930-1960 – Fernseher

Ein Pionier in unserem Land war Jaroslav Šafránek (außerordentlicher Professor für Experimentalphysik an der Karlsuniversität in Prag) in den 1930er Jahren. Im Jahre 1935 stellte er den ersten Fernsehempfangsapparat in der Tschechoslowakei fertig. Das große Interesse am Fernsehen wurde von tschechoslowakischen Funkamateuren geweckt.

Die Kraft der Befragung setzt sich fort.

1939 - Radioübertragung an das Protektorat (alle Mitarbeiter jüdischer Herkunft müssen gehen).

Film und der Lack Propaganda

Es gab eine Reihe von Filmen, die die territoriale Expansion des Dritten Reiches darstellten. Diese Filme hatten keinen großen künstlerischen Wert.

Medien und Kommunismus

1948 stand die CTK unter kommunistischer Diktatur. Sie dient als Instrument der politischen Propaganda der herrschenden Partei. Es gibt eine starke Zensur, die Verstaatlichung des Radios.

Zwei Arten der Berichterstattung: für die Öffentlichkeit und Nichtöffentlichkeit (für hohe Partei- und Staatsbeamte). Die Agentur ist formal der Regierung unterstellt, wird aber

tatsächlich vom Zentralkomitee der Kommunistischen Partei der Tschechoslowakei verwaltet.

50er Jahre - Verschärfter ideologischer Krieg

In den nächsten 40 Jahren haben die Medien in der Tschechoslowakei begonnen, "dem Volk und der Kommunistischen Partei" zu dienen.

1952 - Gründung der Hauptpressestelle Verwaltung.

1952 - Der Tschechoslowakische Rundfunk begann, die Sendungen des Freien Europas abzusetzen (Beginn 1950).

Medien vor der Besetzung von 1968

Radio: natürliche Zivilsprache, Kritik, Offenheit, Empfang von Weltfunksendern - Erweiterung des Angebots, aber Schwächung der staatlichen Kontrolle.

Fernsehen: nur ein Programm, eine enorme Zunahme von Konzessionären, "Zucker" - Abschluss des Baus eines grundlegenden Netzwerks von Sendern.

1968 Beruf - Zeit der Normalisierung

ROZHLAS: Störsender ausländischer Sender (Free Europe, Voice of America).

FERNSEHEN: Sender für das zweite Programm, 1973 - Erste Farbsendung, Personenreinigung, Erneuerung der Zensur, ideologische Konzeption, Angriffe auf Dissens.

DRUCK: Illegale Veröffentlichung von Zeitungen und Zeitschriften.

80er Jahre und andere

1986 - Tschernobyl.

1989 - Die Samtene Revolution.

Eine große Veränderung ist die Entwicklung von Personalcomputern?

1981 - Computer der 4. Generation.

1990-2017

Anfang der 90er Jahre - Medientransformation, Ende der Zensur - steuern die Medien auf das persönliche Eigentum zu, die Medien sind eine Institution der Meinungsfreiheit, ein Forum zur Diskussion von Fragen des öffentlichen Interesses und der Privatwirtschaft.
1991 - Einführung des ersten Webbrowsers - WorldWideWeb am CERN.

EINFÜHRUNG IN DIE PROGRAMMIERUNG MIT JAVA

1. Erstellen von Variablen

Werte werden in Variablen abgespeichert. Eine Variable ist eine mit einer Bezeichnung versehene Stelle in einem Speicher. Im Rahmen der Variablenerstellung ist dies ihre Deklaration. Für jede Variable in der Programmiersprache JAVA muss zuerst festgelegt werden, welcher Datentyp darin enthalten sein soll (in diesem Fall, siehe Beispiel 1 – int, boolean, char). Ihre Bedeutung ist im nächsten Absatz erklärt.

Im Anschluss daran werden für die Variablen Bezeichnungen ohne diakritische Zeichen oder Leer-zeichen vergeben. Die Leerräume zwischen den Wörtern werden einfach ausgelassen, jedes andere Wort beginnt mit einem Großbuchstaben und der Begriff wird mit einem Semikolon abgeschlossen.

Beispiel 1

```
int a;
boolean IsTrue;
char CarConsumption;
```

In Beispiel 1 wurden Variablen erstellt, welchen jedoch noch kein Wert zugewiesen wurde. In Beispiel 2 hingegen wurden Variablen erstellt, welche bestimmten Werte zugewiesen wurden. Die erste Zuordnung von Werten wird Initialisierung genannt.

Beispiel 2

```
int a=2;
boolean IsTrue =true;
char CarConsumption ='A';
```

Möchte man mehrere Variablen gleichzeitig deklarieren, muss man diese durch ein Komma trennen:

```
int a, b;
```

Es wird zwischen einem Ausdruck und einem Befehl unterschieden. Ein Ausdruck hat immer einen Wert, welcher durch die Auswertung des Ausdrucks gewonnen wird. Somit sind z.B. 42 und x+1 Aus-drücke. Ein Befehl ist ein Code, der etwas macht, z.B. X=1; hierbei handelt es sich um einen Befehl, welcher der Variable x den Wert 1 zuordnet. Wenn man einen Ausdruck hat, der etwas evaluiert, kann dieser in der Regel durch das Anhängen eines Semikolons befohlen werden. In diesem Fall sagt man, dass die Evaluation des Ausdrucks einen Nebeneffekt hat.

2. Datentypen

2.1. Integer

Int: Man kann nur ganze Zahlen eingeben, welche in einem Spektrum zwischen -2,147,483,648 und 2,147,483,647 liegen, inklusive Null. Ein Integer kann z.B. der Anzahl von Computeroperationen pro Sekunde entsprechen.

Andere Integer-Datentypen sind in der Tabelle aufgelistet:

Art	Bytes	Spektrum	
byte	1	-128	127
short	2	-32 768	32 767
int	4	-2147 483 648	2147 483 647
long	8	-9,22 x 1018	9,22 x 1018

Liegt das Ergebnis einer Operation nicht innerhalb des erlaubten Intervalls eines bestimmten Datentyps, tritt ein Überlauf auf. In diesem Fall ist das Ergebnis das Gegenteil des Ergebnisses der mathematischen Operation. Es führt zwar zu keinem Fehler in JAVA, aber es sollte bedacht wer-den, z.B. wenn man einer Byte-Variable den Wert 127 zugewiesen hat und die Zahl 1 addiert, wird die Variable den Wert -128 erhalten.

2.2. Echte Zahlen

Um mit echten Zahlen zu arbeiten, verfügt Java über zwei primitive Datentypen: Float und Double. Beide verwenden dieselbe Darstellungsmethode: Die echte Zahl wird als ein Dreifachzeichen, Mantisse und Hochzahl gespeichert, weshalb Zahlen nur ungefähr abgespeichert werden können. Der Typ Float verwendet 32 Bit: ein Bit deckt ein Zeichen

ab, acht Bit die Hochzahl und 23 Bit die Mantisse. Double verwendet 64 Bit: ein Bit für das Zeichen, elf für die Hochzahl und 52 für die Mantisse.

2.3. Umwandlung

Darunter versteht man die Umwandlung eines Werts in einen anderen Datentyp, zB die Umwandlung eines Double in einen Integer. Manche Umwandlungen werden automatisch vorgenommen, z.B. von einem Integer in ein Double, andere müssen erst beauftragt werden, z.B. jene vom Typ Long in einen Typ int. Nachfolgend wird gezeigt, wie man eine automatische Umwandlung festlegen kann:

```
int a = 100;
long a = b;  // Automatic conversion from int to long
```

Wenn eine Umwandlung benötigt, wird der Zieldatentyp in die Klammer vor dem umgewandelten Wert geschrieben. So sieht z.B. eine Umwandlung von Double in einen Integer wie folgt aus:

```
double d1 = 5.85;
int i1 = (int) d1;
```

Das Umwandlungsergebnis wird der Wert sein, welcher dem Eintrag der ursprünglichen Zahl vor dem Dezimalpunkt entspricht (bei nicht-negativen Werten ist dies der gesamte Teil der Zahl), sprich für Variable i2 wird der Wert 5 betragen.

```
double d2 = -4.99;
int i2 = (int) d2;
```

Für Variable i2 ist der Wert -4. Man muss sich bewusst sein, dass dies nicht der gesamte Teil der Zahl ist, da der gesamte Teil der Zahl -4,99 die Zahl -5 ist.

2.4. Logische

Boolean: Enthält nur die Ausdrücke “wahr“ oder „falsch“. Er wird auch logische Variable genannt. Vergleicht man zwei numerische Variablen in a<b, sollen sie a und b heißen, und wenn a tatsächlich kleiner als b ist, ist der Ausdruck a<b wahr und „wahr“ wird fest-

gelegt und so wird auch a zur Variable c definiert. Befindet sich „wahr“ nicht in c, wird „falsch“ darin gespeichert.

```
int a = 5;
int b = 6;
boolean c = a<b;
System.out.println(c);
```

- In diesem Beispiel schreibt das System “wahr”;
- „Falsch“ wird intern als 0 dargestellt, der Wert „wahr“ als 1.

2.5. Zeichen

Char: Darin lässt sich ein Zeichen speichern. Vor und nach diesem Zeichen muss ein Apostroph (ein einfaches Anführungszeichen) stehen. Es lassen sich Zeichen aus der Unicode-Tabelle drucken. Der char-Wert kann als der Zeichenindex in der Unicode-Tabelle verstanden werden.

```
char c ='H';
System.out.println(c) ;
```

2.6. Zeichenketten

Man verwendet string um mit Zeichenketten zu arbeiten. Die Konstanten der Kette werden in Anführungszeichen geschrieben.

```
String s = ” Hi how are you?”;
```

Ketten können durch den Operator + miteinander verbunden werden.

```
String s1 = ”jdk”, s2 = ”7.0”;
String s3 = s1 + s2;  // Creates a string ”jdk7.0”
```

Der Zeichenkette lässt sich noch ein weiterer Wert hinzufügen. In diesem Fall wird der Wert zuerst in eine Zeichenkette umgewandelt und anschließend werden die Zeichen-

ketten zusammengeführt.

```
int x = 42;
String s = ”answer is ” + x;
```

Das obere Beispiel erzeugt diese Zeichenkette: "answer is 42"

2.7. Kommentare

Es gibt in JAVA drei Arten von Kommentaren

- einzeilige – beginnen mit // und gehen bis an das Ende der Zeile
- mehrzeilige – beginnen mit / * und werden mit den Zeichen * / beendet
- Dokumentation – beginnt mit den Zeichen / ** und endet mit den Zeichen * /

Dokumentationskommentare sind für die Verarbeitung durch Java.doc vorgesehen, welches Dokumentationen im HTML-Format erzeugt.

```
/**
   * Main program class.
   */

   public class Main {
         public static void main( String[] args ) {
               /* Prints the answer */
               System.out.println( 42 ); // answer is 42
         }
   }
```

Mithilfe von Kommentaren werden im Quelltext zusätzliche Informationen ergänzt. Der Transporter überspringt Kommentare, weshalb sie keine Auswirkungen auf die Programmausführung haben. Kommentiert werden sollten wichtige Variablen, ungewöhnliche Verfahren und nicht-traditionelle Algorithmen, sprich Seiten, deren Bedeutung sich den Lesern nicht auf den ersten Blick erschließt.

2.8. Bildschirmeingabe und - Ausgabe

In diesem Kapitel wird gezeigt, wie man etwas mit der Tastatur etwas auf den Bildschirm

schreibt sowie dort liest. Die Bildschirmausgabe ist die sogenannte Output Current (System.out). Es werden hierfür drei Methoden verwendet: print (), println () und printf (). Der Abruf unten zeigt Hi Baby an.

```
System.out.println( ” Hi Baby!” );
```

Die Methoden print () und println () drucken den Wert aus, welcher in den Klammern nach der Bezeichnung der Methode festgelegt wurde (dieser Wert wird Parameter genannt). Er weicht durch das Hinzufügen eines Übergangs zu einer neuen println ()-Zeile ab, weshalb der nächste Abruf von print () oder println () vom Anfang der Zeile an ausgeben wird. Bei der Ausgabe kann man Zeichenketten mithilfe des + Operators dazu bringen, sich aneinander anzuschließen.

```
int v = 200;
System.out.println( ”Car moved ” + v + ” km/h” );
```

Eine elegantere Ausgabe wird durch die Methode printf () ermöglicht (die sogenannte formatierte Ausgabe), welche der Sprache C ähnlich ist.

```
int m = 6;
System.out.printf( ” African elephant weighs %d tons”, m );
```

Die printf () Parameter sind eine Formatierungszeichenkette und eine Liste von Werten. Der Formatierungsstring kann die sogenannten Ausgabeumwandlungen enthalten, welche die Form fest-legen, in welcher der passende Wert ausgegeben wird. Jede Umwandlung beginnt mit einem %-Zeichen, so entspricht zB %D der Ausgabe des Dezimal-Integers. Wenn der Befehl ausgeführt wird, wird der passende Wert aus der Werteliste anstelle der Umwandlung eingesetzt. Fehlt der Wert oder passt er nicht mit der Ausgabeumwandlung zusammen, tritt ein Fehler auf.

Ein spezieller Fall ist eine %n-Umwandlung, die mit keinem Wert in der Werteliste übereinstimmt. Diese Umwandlung wird durch eine Verschiebung in eine neue Zeile wiedergegeben. In MS Windows wird das Zeichenpaar \ r (Zellenrückgabe) und \ n (Zeileneingabe) zur Verschiebung in eine neue Zeile verwendet. Unix-Systeme verwenden \n. Die %n-Umwandlung stellt sicher, dass der richtige Übergang zur neuen Zeile verwendet wird, sprich \ r \ n auf MS Windows \ n auf Unix.

Für echte Zahlen gibt es eine %f-Umwandlung. Hier kann man die Anzahl an Ziffern nach

dem Dezimalpunkt festlegen (der Standardwert ist 6), sprich %. 2f gibt zwei Ziffern nach dem Dezimal-punkt aus.

```
System.out.printf( ” Euler’s constant is about %.2f%n”,
Math.E );
```

Siehe die unter Verwendung der %c-Umwandlung ausgegebenen Zeichen:

```
char c = ’@’;
int i = c;
System.out.printf( ”Character ’%c’ Is in the Unicode table
in position %d%n”, c, i );
```

Für Strings wird eine %s-Umwandlung verwendet:

```
String Java = ”Java”;
System.out.printf( ” Our favorite programming language
is %s%n”, Java );
```

Zur Ablesung von der Tastatur gibt es in JAVA einen sogenannten „Input Stream“ (System.in). Dieser wird zumeist nicht direkt, aber durch die Scanner-Klasse verwendet. Diese Klasse bietet Methoden zum Ablesen einfacher Typen und Strings. Wenn die Scanner-Klasse zum Einsatz kommt, beginnt das Programm für gewöhnlich mit einem wichtigen Befehl, welcher dem Übersetzer sagt, wo er nach der Scanner-Klasse suchen soll. Vor dem ersten Ablesen wird mithilfe des neuen Schlüsselworts eine Instanz der Scanner-Klasse erstellt. Um einen Integer-Wert wiederzugeben, muss man die nextInt ()-Methode bei dieser Instanz abrufen.

```
import Java.util.Scanner;
   public class Read {
      public static void main( String[] args ) {
           Scanner sc = new Scanner( System.in );
           int x = sc.nextInt();
           System.out.printf( ” Readed value is: %d%n”, x );
      }
   }
```

Die Methode nextDouble () wird verwendet, um einen Double-Wert wiederzugeben. Gelesen wird der String durch next ().

```
Scanner sc = new Scanner( System.in );
double d = sc.nextDouble();
System.out.printf( ”Readed number is: %f%n”, d );
String s = sc.next();
System.out.printf( ”Readed string is: %s%n”, s );
```

3. Operatoren

Zahlen können gängige arithmetische Operationen in Java ausführen: Addition, Subtraktion, Multiplikation, Division und Modulo (der Rest nach einer Integer-Division). Operatoren werden verwendet, um Operationen zu schreiben. So wird z.B. die Addition mit dem Operator + und der Modulo mit dem Operator % geschrieben. Daher wird mit dem Eintrag x + 2 eine Additions-Operation geschrieben. Jeder Operator funktioniert mit einem oder mehreren Werten oder Variablen, welche Operanden genannt werden. Je nach Anzahl der Operanden unterscheidet man zwischen unären (ein Operand), binären (mit zwei Operanden) und ternären Operatoren (mit drei Operanden). Der föderative Operator ist zB ++ (Erhöhung) und ein binärer Operator ist z.B. = (Zuordnung). Der einzige ternäre Operator ist ? (Bedingungsoperator). Das Ergebnis der Operatorausführung ist ein bestimmter Wert (man sagt, dass der Operator den Wert zurückgibt), z.B. liefert ein binärer Operator + (Addition) die Summe seiner Operanden. Der Ergebnistyp hängt vom Operator und manchmal auch von den Operanden ab. Geben Operatoren einen Integer-Wert zurück, ist das Ergebnis entweder int oder long. Fügt man z.B. zwei int-Werte hinzu, wird das Ergebnis ein int sein, fügt man zwei long-Werte hinzu, wird das Ergebnis long sein und fügt man Zwei-Byte-Werte hinzu, wird das Ergebnis ein int sein. Ein Operator / (aufgeteilt) gibt Integer-Anteile für Integer-Operanden zurück, z.B. 15/4 entspricht 3 und -15/4 ist -3.

Wenn der Wert des zweiten Operanden 0 ist, tritt ein Fehler auf. Wenn zumindest ein Operand eine echte echte Zahl ist (z.B. Float oder Double), wird der zweite Operand in eine echte Zahl umgewandelt und der dividierte Operator wird das Verhältnis beider Operanden zurückgeben, z.B. 4,5/3 ist 1,5. Ist der zweite Operand 0, wird die echte Division zu einem dieser Werte führen: Plus unendlich, wenn der erste Operand positiv ist, minus unendlich, wenn der zweite Operand negativ ist oder NaN (Not a Number), wenn der erste Operant positiv unendlich, negativ unendlich, NaN oder 0 ist. Der Operator % (Modulo) gibt den Rest nach einer Integer-Division zurück, zB 17% 4 ist 1. Die-ser Operator gilt auch für negative Werte, unterscheidet sich jedoch von der Mathematik. Das Vorzeichen des Ergebnisses ist immer dasselbe wie jenes des ersten Operanden: -17% 4 ist -1, 17% -4 ist 1 und -17% -4 ist 1.

Operatoren kann man auch erhalten, sprich mehrere Operatoren können geschrieben werden. So ist z.B. X +2 *y ein Ausdruck, welcher Additions- und Multikplikationsoperatoren enthält. Die Rang-ordnung von Operatoren ist die Grundlage für die Bewertung der Operatoren-Priorität (Präzedenz), z.B. im Ausdruck x + 2 * y werden zunächst die Multiplikation und dann die Addition ausgeführt, da der Multiplikationsoperator eine höhere Priorität als der Additionsoperator hat. Eine andere Rangordnung kann in Klammern festgelegt werden: (x + 2) * y. Wenn mehrere Operatoren mit derselben Priorität in einem Ausdruck verwendet werden, wird die Assoziativität des Operators durch die Evaluationsreihenfolge festgelegt, z.B. in x – y – z, wird zuerst x – y berechnet, ehe z vom Ergebnis abgezogen wird. Man sagt, dass der Subtraktionsoperator von links nach rechts geht. Aus diesem Grund hat der Ausdruck x – y – z denselben Wert wie der Aus-

druck (x – y) – z.

Einige Operatoren arbeiten anders herum, sprich von rechts nach links, wobei der Zuschreibungs-operator eine Ausnahme bildet. Der zurückgegebene Wert des Operators ist der Wert des linken Operanden nach der Zuordnung. Im Ausdruck x = y = 1 ist die erste Zuordnung y = 1 und der zu-rückgegebene Operatorwert (in diesem Fall 1) wird x zugeordnet. Deshalb wird der Ausdruck x = y = 1 als x = (y = 1) gewertet.

3.1. Steigernde und verringernde Operatoren

Der Erhöhungsoperator (++) ist dafür verantwortlich, dass der Wert einer Variable um 1 erhöht wird. Er kann auf zwei Arten geschrieben werden: als Präfix oder als Suffix. In der Präfix-Notation steht der Operator vor dem Operanden, in der Suffix-Notation dahinter. In beiden Fällen wird die Variable erhöht, allerdings gibt es einen Unterschied betreffend den Wiedergabewert des Opera-tors. Der Präfix-Operator gibt den Wert der Variable nach der Vergrößerung zurück und der Suffix-Operator den Wert vor der Vergrößerung.

```
int x = 1;
int y = ++x;
```

Im Ausdruck y = ++ x wird der Erhöhungsoperator zuerst ausgeführt, da er eine höhere Priorität als der Zuordnungsoperator hat. Dadurch erhöht sich der Wert x um 1. Der zurückgegebene Wert des Operators ist x nach der Vergrößerung, sprich 2. Dieser wird als ein Zuordnungsoperator verwendet. Y wird daher 2 sein.

```
int x = 1;
int y = x++;
```

Im Ausdruck y = x++ wird der Erhöhungsoperator zuerst ausgeführt. Sein Rückgabewert ist der Wert der x-Variable vor der Vergrößerung, sprich 1. Der Wert y wird in der Variable y gespeichert. Der Verringerungsoperator (-) sorgt dafür, dass sich der Wert der Variable um 1 verringert. Er wird ähnlich wie der Erhöhungsoperator verwendet.

```
int x = 1;
System.out.println( --x );
```

3.2. Logische Operatoren

Logische Ausdrücke können mittels logischer Operatoren aneinandergereiht werden. Der logische Operator hat boolesche Operanden und gibt einen booleschen Wert zurück. Es gibt zwei logische Operatoren. Der Operator „und“ && gibt den Wert „wahr“ dann und nur dann zurück, wenn beide Operanden wahr sind.

```
if( x == 0 && y == 0 ) {
      system.out.println( ”x and y are equal to zero” );
  }
```

Der Operator „oder“ || gibt den Wert „wahr“ zurück, wenn zumindest einer der Operanden wahr ist.

```
if( x == 0 || y == 0 ) {
      System.out.println( ” At least one of the numbers x,
      y is equal 0” );
}
```

Beide Operatoren verwenden die sogenannte abgekürzte Evaluation, sprich der zweite Operand wird nur dann ausgewertet, wenn der Wert des gesamten Ausdrucks nach der Auswertung des ersten Operanden nicht bekannt ist. Hat der erste Operand einen falschen Wert im logischen Produkt, wird der zweite Operand gar nicht erst ausgewertet und der Wert des gesamten Ausdrucks ist falsch. Da diese Operatoren oft in Bedingungen verwendet werden, sind sie konditional.

Neben Bedingungsoperatoren hat Java auch Operatoren, die immer beide Operanden evaluieren. A & (logisches Produkt) wird „und“ | (logische Summe) geschrieben. Sie können an derselben Stelle wie Bedingungsoperatoren verwendet werden.

```
boolean b1 = x > 0 & y == 1;
boolean b2 = x <= 0 | y <= 0;
```

Kombiniert man „und" und „oder", ist es notwendig zu bedenken, dass „und" eine höhere Priorität als „oder" hat:

```
if( x == 0 || y > 0 && z > 0 ) {
      System.out.println( "x is zero or y and z are positive " );
  }
```

3.3. Zuschreibungsoperatoren

Einer der Zuschreibungsoperatoren, nämlich der Operator =, ist bereits bekannt. Die anderen Zuschreibungsoperatoren ermögliche es, besondere arithmetische Operationen mit Variablen auszuführen, zB der Operator + = fügt der Variable einen zweiten Operanden hinzu.

```
    x += 5;
```

Andere Zuschreibungsoperatoren sind - =, * =, / =, % =. Jeder davon ist eine Aufzeichnung einer damit übereinstimmenden Operation mit der jeweils links befindlichen Variablen. So führt z.B. der Operator % = eine Modulo-Operation aus:

```
    x %= 6;   // dasselbe wie x = x % 6
```

3.4. Priorität von Operatoren

Alle Operatoren geben einen Wert zurück, sprich ein Operationsergebnis. Sie alle haben dieselbe Priorität und gehen von rechts nach links. In der Aussage

```
    int y = x + = 1;
```

führt der Operator zunächst = = aus und erst dann =. Der + Operator = gibt den Wert x zurück, nachdem er um 1 erhöht wurde. Dieser Wert wird als Wert für den zweiten Operator = verwendet.

Die untersuchten Operatoren lassen sich nach Priorität (von der höchsten zur geringsten) sortieren:

- erhöhend (++), verringernd (-)
- verteilend
- Multiplikation (*), Division (/), Modulo (%)
- Addition (+), Subtraction (-)
- Zuschreibungsoperatoren (=, + =, - =, * =, / =,% =)

Die Priorität legt fest, wie stark Operatoren Operanden gewichten. So hat zB die Verteilung eine höhere Priorität als die Multiplikation, weshalb beim Ausdruck (int) d * 2 zuerst die Verteilung und dann erst die Multiplikation vorgenommen wird. Das Ergebnis ist 10.

```
double d = 5.8;
int i = (int) d * 2;  // same like ((int) d) * 2
System.out.println( i );
```

Zuschreibungsoperatoren gehen von rechts nach links und arithmetische Operatoren (Addition, Subtraktion, Multiplikation, Modulo) von links nach rechts.

Alle binären Operatoren werten die Operanden in derselben Reihenfolge aus: zuerst links und dann rechts. Diese Reihenfolge ist von Bedeutung, wenn die Operanden-Evaluation einen Neben-effekt hat.

```
int x = 0;
int y = x + x++;
```

In der zweiten Zeile wird der Operator + zuerst ausgewertet. Dessen linker Operand ist 0, weshalb der rechte Operand auch 0 ist. Darum gibt der Operator 0 zurück und ordnet den Wert y zu. Wird der rechte Operand ausgewertet, wird die Variable x 1 erhöht (die Evaluation hat einen Nebeneffekt). Verändert sich die Reihenfolge der Operanden, wird y der Wert 1 zugeordnet.

```
int x = 0;
int y = x++ + x;
```

3.5. Relationale Operatoren und Gleichheitsoperatoren

Um Bedingungen zu formulieren, verwendet man sogenannte relationale und Gleichheitsoperatoren. Relationale Operatoren sind:

- kleiner als (<)
- größer als (>)
- kleiner gleich (<=)
- größer gleich (> =)

Gleichheitsoperatoren sind:

- ist gleich (==)
- ist nicht gleich (! =)

4. Grundlegende Programmierstrukturen

4.1. IF

Die Befehle “if” und “switch” werden verwendet, um das Programm zu verzweigen. “If”-Statements erlauben es, ein Programm durch eine Bedingung abzubrechen. Sie starten mit dem Schlüsselwort „if“, gefolgt von booleschen Operatoren in Klammern. Boolesch ist ein Ausdruck, dessen Wert wahr oder falsch ist. Es folgt der eigentliche Befehl. Wird dieser ausgeführt, wird der Ausdruck in den Klammern ausgewertet und ist dieser wahr (wird die Bedingung erfüllt), wird der Befehl ausgeführt. Im nachfolgenden Beispiel wird getestet, ob die Variable x null ist. Wenn diese gleich null ist, wird der x-Variable der Wert 2 zugeordnet.

```
if (x==0) x=2;
```

Man verwendet sowohl relationale als auch Gleichheitsoperatoren, um Bedingungen zu formulieren. Ein “If”-Statement kann auch den “else”-Zweig enthalten, welcher ausgeführt wird, wenn die Bedingung nicht erfüllt wird. Wird die Bedingung nicht erfüllt und fehlt ein alternativer Zweig, wird nichts geschehen (es wird nach dem „if“-Statement fortgesetzt).

```
if( x == 0 )
      System.out.println( ” Can not be divided by zero!” );
else
      z=5/x;
```

Möchte man mehrere Befehle schreiben, wird ein Block verwendet. Der Block beginnt mit einer öffnenden Klammer {und endet mit einer schließenden Klammer}. Ein Block ist ein Java-Befehl, weshalb man ihn in Operationen verwenden kann, welche einen Befehl enthalten:

```
if( x == y ) {
      x++;
      y--;
}
```

Ein Block begrenzt die Validität der Variablendeklaration. Jede Deklaration ist nur bis zum Ende des Blocks, in dem sie gelistet ist, gültig. Man spricht von einer Lokalität im Block.

```
if( x == y ) {
      int z = y;
   }  // Diese Klammer beendet die Deklaration der
      //Variable z
      //Hier kann man z nicht verwenden
```

Man kann boolesche Variablen in Klammern verwenden:

```
// In the month variable we have the serial number of the month
boolean isMay = (month == 5);
     if( isMay ) {
          System.out.println( ”time for love” );
     }
```

Um eine gegenteilige Bedingung zu formulieren, wird der logische Negierungsoperator (!) verwendet.

```
boolean isHere = true;
if( ! isHere ) {
      System.out.println( ”Is not here” );
}
```

Soll das Programm vergezweigt werden, z.B. durch den Wert von Integer-Variablen, lässt sich hierfür eine Verkettung von „if“-Statements verwenden.

```
if( x == 1 ) {
      System.out.println( ”one” );
} else if( x == 2 ) {
      System.out.println( ”two” );
} else if( x == 3 ) {
      System.out.println( ”three” );
}
```

4.2. Switch

Daraus ergibt sich, dass wenn „if“-Statements wie im vorherigen Beispiel aneinandergereiht wer-den, diese manchmal durch den Befehl „switch“ ersetzt werden können. Dessen Formulierung beginnt mit dem Schlüsselwort „switch“. Es folgt ein Integer- oder String-Datentyp (oder ein Daten-typ, der in einen Integer umgewandelt werden kann) in Klammern sowie ein Block mit einer beliebigen Anzahl an Fall-Kennsätzen. In jedem Fall gibt es eine Konstante (oder einen konstanten Aus-druck, einen Ausdruck, dessen Wert zur Übersetzung dient), ein Semikolon und eine Befehlssequenz.

```
switch( x ) {
      case 1:
            System.out.println( ”one” );
            break;
      case 2:
            System.out.println( ”two” );
            break;
      case 3:
            System.out.println( ”three” );
}
```

Bei der Ausführung wird der Ausdruck als das Schlüsselwort “switch” angesehen und dessen Wert wird mit jenen Werten verglichen, die in so einem Fall vorgegeben sind (in der Reihenfolge, in der sie aufgelistet sind). Einmal angefangen, starten die Befehle und werden Ausführung endet mit dem „break“-Befehl. Fehlt ein solcher „break“-Befehl, werden alle Befehle ausgeführt, bis das En-de des Switch-Befehls erreicht ist. Ein „switch“-Befehl kann einen Standardzweig enthalten, welcher ausgeführt wird, wenn kein Fall-Kennsatz passt.

5. Zyklen

Zyklen werden verwendet, um Befehle erneut auszuführen.

5.1. While

Der While-Zyklus beginnt mit dem Schlüsselwort “while”, auf welches die Bedingung und der Zyk-lus-Körper in Klammern folgen. Der Körper eines Zyklus ist entweder ein Befehl oder ein Block. Innerhalb eines „while“-Zyklus wird die Bedingung überprüft und wenn sie erfüllt wird, wird der Zyklus-Körper ausgeführt. Anschließend wird die Bedingung erneut überprüft und, wenn sie zu-trifft, wird der Zyklus-Körper erneut ausgeführt usw. Wird die Bedingung nicht erfüllt, wird mit den Befehlen anderer Zyklen fortgesetzt. Wird die Bedingung bereits zu Beginn nicht erfüllt, wird der Zyklus-Körper nicht ein einziges Mal ausgeführt. Der „while“-Zyklus ist also ein Zyklus mit einer Wiederholungsrate von 0 oder mehr.

Beispiel einer Syntax:

```
While(condition) {
// body
}
```

Ein konkretes Beispiel

```
int x = 5;
while( x > 0 ) {  // Do until x is greater than zero
      System.out.println( x );
      x --;
}
```

Dieses Beispiel schreibt Zahlen von 5 bis 1. Der Zyklus wird deshalb fünfmal wiederholt.

5.2. Do while

Der “do while”-Zyklus beginnt mit dem Schlüsselwort „do“, gefolgt von dem Zyklus-Körper. Im Anschluss daran wird das Schlüsselwort „while“ mit der dazugehörigen Bedingung geschrieben, wie das Syntax-Beispiel zeigt. Die Abfolge sieht wie folgt aus: Zunächst wird der Zyklus-Körper ausgeführt, die Bedingung wird überprüft. Wird sie erfüllt, wird der Zyklus-Körper erneut ausgeführt, gefolgt von der Überprüfung der Bedingung usw. Wird die Bedingung nicht erfüllt, wird nach dem Zyklus fortgesetzt. Der Zyklus-Körper ist immer zumindest einmal „do“.

Beispiel einer Syntax:

```
do  {
// body
} while (condition);
```

Ein konkretes Beispiel

```
do  {
System.out.println( x );
      x --;
} while (x > 0);
```

5.3. For

Der “for”-Zyklus hat diese Gestalt: for (Zyklus-Variable, Bedingung, Befehl) und wird wie folgt um-gesetzt: Zunächst wird der Kontrollvariablenzyklus initialisiert, anschließend die Bedingung über-prüft und wenn diese erfüllt wird, wird der Zyklus-Körper ausgeführt. Anschließend wird der Befehl ausgeführt, die Bedingung erneut überprüft und, sofern zutreffend, wird der Zyklus-Körper erneut ausgeführt usw. Die Initialisierung der Kontrollvariable des Zyklus erfolgt nur einmal ganz zu Beginn. Wird die Bedingung zu Beginn nicht erfüllt, wird der Zyklus-Körper kein einzige Mal ausgeführt.

Beispiel einer Syntax:

```
for (cycle variable; condition; command){
// body
}
```

Ein konkretes Beispiel:

```
int a;
for( a = 1; a < 10; a++ ) {
      System.out.println( a );
}
```

Die Zyklus-Variable kann innerhalb des Zyklus neu deklariert werden. Diese Deklaration ist nur in diesem bestimmten Zyklus gültig (sprich im Header und im Zyklus-Körper). Man spricht davon, dass die Variable in diesem Zyklus lokal ist.

```
for( int a = 1; a <= 10; a++ ) {
      System.out.println( a * a );
}
```

Es ist kein Problem, wenn jene Teile, welche die Zyklus-Variable, Bedingung oder den Befehl kontrollieren, fehlen. Fehlt jedoch die Bedingung, wird der Zyklus unendlich (da die Bedingung stets erfüllt wird). Wenn es keinen Kontroll-Zyklus oder keine Befehlsvariable gibt, wird kein Befehl aus-geführt.

5.4. Abbrechen und fortsetzen

Im Zyklus-Körper kann man einen “break”- und einen “continue”-Befehl verwenden. Der “break”-Befehl beendet sofort die Ausführung des Zyklus.

```
int s = 100;
while( s > 0 ) {
      int n = sc.nextInt();
      if( n == 0 ) {
            break;
      }
      s -= n;
      System.out.println( s );
}
// Here will continue after the break executed
```

Der “continue”-Befehl beendet die Ausführung des Zyklus-Körpers und springt zur Zyklus-Bedingung.

```
int s = 0;
do {
      int n = sc.nextInt();
      if( n == 0 ) {
            continue;
      }
      s += n;
      System.out.println( s );
      // here goes continue
} while( s < 100 );
```

6. Statistische Methoden

Bis jetzt wurde das gesamte Programm in die Hauptmethode geschrieben. Wenn dieselbe Befehls-sequenz an mehreren Stellen ausgeführt werden soll, muss man diese Befehle wiederholen. Das kann vermieden werden. Methoden ermöglichen es, den Code in logische Einheiten zu unterteilen und diese wiederzuverwenden. In diesem Kapitel geht es darum, Methoden als statische Methode zu verstehen. Eine besondere statische Methode ist bereits bekannt: Es handelt sich um die zuvor genannte Hauptmethode. Neben dieser Hauptmethode lassen sich andere Methoden in Klassen deklarieren, wie z.B. die Methode zur Ausgabe von Programminformationen.

```
class MainClass {
      static void printInfo() {
            System.out.println( "Version: 1.0" );
            System.out.println( " Autor: 007" );
      }
      public static void main( String[] args ) {
            printInfo ();
      }
}
```

Die Deklaration einer statischen Methode beginnt mit dem statischen Schlüsselwort (siehe Beispiel oben), gefolgt vom Wiedergabetyp und der Bezeichnung der Methode. Der Wiedergabetyp kann ein beliebiger Java-Typ sein. Gibt die Methode keinen Wert zurück, wird der Wiedergabetyp als leer festgelegt. Die Bezeichnung der Methode beginnt für gewöhnlich mit einem Kleinbuchstaben. Besteht die Bezeichnung aus mehreren Wörtern, werden die einzelnen Wörter durch Großschreibung des jeweils ersten Wortbuchstaben kenntlich gemacht, z.B. SpoctiPolomerCurzniceOpsane. Es ist nicht üblich, Unterstrichen in Namen zu verwenden. Auf die Methodenbezeichnung folgt in Klammern eine Parameter-Liste. Die Parameter-Liste besteht aus Deklarationen von Variablen. Um die Deklarationen in der Parameter-Liste voneinander zu trennen, wird ein Komma eingesetzt.

6.1. Abrufmethoden

Der Methodenabruf wird an jene Stelle geschrieben, wo die Methode ausgeführt werden soll. Der Abruf einer Methode besteht aus der Methodenbezeichnung und der Parameter-Liste. Der Rückgabewert der Methode wird der Wert des Ausdrucks sein, welcher nach dem Wiedergabe-Statement festgelegt ist.

Die Reihenfolge, in welcher die Methoden deklariert sind, ist unerheblich. Man kann auch eine Methode abrufen, welche erst später deklariert ist.

```
class MainClass {
    public static void main( String[] args ) {
        Scanner sc = new Scanner( System.in );
        int x = sc.nextInt();
        long factorial = countFactorial( x );
        System.out.printf( "%d! = %d%n", x, faktorial );
    }
    static long countFactorial ( int n ) {
        long fact = 1;
        for( ; n > 1; n-- ) {
            fact *= n;
        }
        return fact;
    }
}
```

In der leeren Methode kann man das Wiedergabe-Statement ohne Parameter nutzen. Dieses kommt beim vorzeitigen Beenden der Methode zum Einsatz.

```
// print rectangle a x b from @
static void printRectangle ( int a, int b ) {
       // The minimum rectangle side size is 2
      if( a < 2 || b < 2 ) {
            return;
      }
      for( ; a > 0; a-- ) {
            for( int i = 0; i < b; i++ ) {
                  System.out.print( '@' );
            }
            System.out.pritln();
      }
}
```

Das Wiedergabe-Statement kann in der Methode mehrfach vorkommen. Es wird jedoch immer einmalig als letzter Befehl der Methode ausgeführt. Mit der Ausführung des Wiedergabe-Statements wird die Methode sofort beendet.

```
static boolean isPrimeNumber ( int n ) {
       if( n == 2 ) {  // 2 prime number
            return true;
       }
       if( n % 2 == 0 ) {
            // Even number is not a prime number (except 2)
            return false;
       }
       int sqrt = (int) Math.sqrt( n );
            for( int i = 3; i <= sqrt; i += 2 ) {
            if( n % i == 0 ) {
              // We found a divisor, so n is not a prime number
              return false;
            }
       }
       return true;
}
```

7. Instanzvariablen

Instanzvariablen werden auch als Instanzattribute oder kürzere Attribute (Instanzen von Instanzen oder Arrays) bezeichnet. Die statischen Methoden sind bekannt und wurden mithilfe des statischen Schlüsselworts bereits deklariert.

Instanziierte Methoden werden auf eine Weise deklariert, die statischen Methoden recht ähnlich ist. Anders als bei statischen Methoden enthält deren Header jedoch kein statisches Schlüsselwort. Darüber hinaus arbeiten instanziierte Methoden oft mit den Instanzattributen

8. Arrays

Ein Array ermöglicht es, mit mehreren Werten desselben Datentyps zu arbeiten. So kann man z.B., um 20 Integer-Werte zu speichern, entweder 20 Variablen eingeben oder 20 Elemente umfassen-de Arrays erstellen. In vielen Fällen gestaltet sich die Arbeit mit einem Array einfacher. Eingegeben wird der Array-Typ in Java mittels eckiger Klammern. Die Deklaration des Variablentyp-Arrays für Integer-Objekte sieht wie folgt aus:

```
int[] p;
```

Ein Array-Typ einer Variable ist die sogenannte Referenz. Sie wird einen Verweis auf das Array ent-halten, die Array-Deklaration selbst jedoch nicht. Ein Array lässt sich mithilfe des Schlüsselworts „new“ erstellen:

```
p = new int[6];
```

In oberen Fall wurde ein Array für sechs Integer-Elemente erstellt. Benötigt man eine Anzahl an Array-Elementen, verwendet man *ArrayBezeichnung.length*, sprich in diesem Fall

```
p.length
```

Der Variablen-Wert wird beim Erstellen eines Arrays festgelegt und kann nicht geändert werden (nur gelesen).

```
System.out.printf( ”array p has %d elemets%n”, p.length );
```

Zugriff auf die einzelnen Elemente des Arrays erhält man mithilfe von Indizes, welche in eckige Klammern geschrieben werden:

```
p[1] = 5;
```

Indizes beginnen immer bei null, die erste Zahl wird den Index 0, die zweite den Index 1, die dritte den Index 2 haben usw. Die Validität eines Index wird immer während der

Laufzeit überprüft. Verwendet man einen ungültigen Index, führt dies zu einem Laufzeitfehler. Einmal erstellt, werden Array-Elemente zu Werten initialisiert, deren interne Darstellung 0 entspricht. Bei numerischen Typen ist das die Zahl 0, bei booleschen Typen ist es „falsch“ und beim Zeichentyp char entspricht dies jenem Zeichen, welches in der Unicode-Tabelle an der Position 0 steht. Man verwendet die „for“-Schleife, um Werte im Feld auszulesen:

```
int[] p = new int[10];
        for( int i = 0; i < a.length; i++ ) {
              p[i] = i;
}
```

Mit der “for”-Schleife wird das meiste der Feldelement-Liste erneut ausgeführt:

```
for( int i = 0; i < a.length; i++ ) {
        System.out.println( a[i] );
}
```

Die Feldlänge darf nicht negative sein. Der Versuch, ein Feld mit negativer Länge zu erstellen, führt zu einem Fehler. Die Erstellung eines speziellen Felds kann mit einer Initialisierung kombiniert wer-den. In diesem Fall wird „new“ nicht verwendet.

```
int[]numbers = { 3, 5, 6, 7};
```

Die Feldgröße wird durch die Anzahl an Werten in den Verbindungsklammern vorgegeben. Das Array kann ein Parameter der Methode und auch ein Wiedergabetyp sein.

8.1. Multidimensionale Arrays

Bis jetzt wurde ein Index verwendet, um das Feldelement festzulegen, welches eindimensionales Feld genannt wird. Java erlaubt jedoch auch das Deklarieren und Erstellen multidimensionaler Arrays. So sieht z.B. ein zweidimensionales Feld aus Integer-Objekten wie folgt aus:

```
int[][] p;
```

Ein multidimensionales Feld in Java ist ein Feld aus Feldern. Die Variable p ist ein Verweis auf ein Array, dessen Elemente eindimensionale Integer-Arrays sind. Es ist möglich, mithilfe des Schlüssel-worts „new“ ein neues Feld zu erstellen.

```
p = new int[2][3];
```

Der Zugriff auf die Feldelemente erfolgt über Indizes:

```
p[0][1] = 1;
```

Die Anzahl von Array-Elementen wird durch die Variablenfeldlänge angegeben:

```
System.out.printf( ”pole p má d řádkůn”, p.length );
System.out.printf( ”první řádek má d sloupcůn”, p[0].length );
```

Man verwendet verschachtelte Zyklen, wenn man mit mehrdimensionalen Arrays arbeitet:

```
for( int i = 0; i < p.length; i++ ) {
        for( int j = 0; j < p[i].length; j++ ) {
             p[i][j] = 1;
        }
}
```

Multidimensionale Arrays können sequentiell erstellt werden. Die Sub-Felder können eine andere Anzahl an Elementen haben. Deshalb ist ein zweidimensionales Array nicht zwangsläufig rechteckig.

9. Klassen

Eine Klasse ist ein Format, welches einen eindeutig eingeschränkten Datensatz (oder Datensätze) und damit verbundene Operationen beschreibt. Man verwendet Klassen, um Instanzen zu erstellen, sprich einzelne Objekte, welche die Daten selbst enthalten (für welche die übereinstimmen-den Operationen abgerufen werden).

Als Beispiel sei an dieser Stelle eine Auto-Klasse angeführt. Diese Klasse beschreibt, dass das Auto eine Marke, einen Fahrzeugtyp, ein Alter und einen bestimmten Kilometer-stand hat. Darüber hin-aus enthält sie die Informationsoperation, das object (), welches den Typ, das Alter und den nicht-Kennzeichen-Typ auflistet. Unter Verwendung dieser speziellen Klasse werden individuelle Instanzen erstellt. Im echten Leben würde man diese als spezielle Fahrzeuge beschreiben.

Jede neu erstellte Instanz (Auto) im Programm ordnet weitere Daten (Marke, Typ, Alter und Kilo-meterstand) zu. Wird die Operatorinformation später abgerufen (), wird dieses Objekt (Instanz) die mit dem Fahrzeug in Verbindung stehende Nachricht zurückgeben.

9.1. Klassendeklaration

Um Klassen zu deklarieren, wird das Schlüsselwort “class” verwendet, welchem ein Zu-gangsspezifikationssymbol vorangestellt ist und auf welches der Klassenname folgt. Der Klassen-Körper selbst ist in einen Block (Klammern) eingebettet. Besteht der Name aus mehreren Wörtern, wird der erste Buchstabe des jeweiligen Worts in Großbuchstaben geschrieben (z.B. ListingInfo). Unterstriche werden nicht verwendet. In der Klasse kön-nen spezifische Variablen und Methoden deklariert werden, welche entweder statisch oder instanziiert sein können.

```
class Cat {
        int weight;  // instance atribut
        int age;
        void showInfo() {  // Instance method
              System.out.println( info );
        }
}
```

Man kann von einer Klasse aus instanziieren. Die Klasse gibt vor, wie Objekte aussehen werden und welche Attribute und Methoden sie haben werden. In diesem Fall wird jede Instanz von „Cat“ zwei Integer-Variablen haben. Durch das Deklarieren der Variable Cat wird eine Variable eingeführt, in welcher sich ein Verweis auf eine Instanz der Klasse Cat speichern lässt.

```
Cat  v;
```

Da sich Variablen des Typs “class” auf Objekte beziehen, werden sie auch Referenzvariablen genannt. Jede Referenzvariable nimmt denselben Platz ein: 32 Bit im 32-Bit JVM und für gewöhnlich 64 Bit im 64-Bit JVM. Andererseits haben Objekte desselben Typs für gewöhnlich eine unterschied-liche Größe. Eine Objektgröße ergibt sich durch dessen Attribute. Objekte werden durch die Verwendung des Schlüsselworts „new“ erstellt:

```
V = new Cat();
```

Nachdem dieser Befehl ausgeführt wurde, wird die Variable v einen Verweis auf eine Instanz der Kubikklasse enthaltene. Auf Attribute und Methoden greift man mittels eines Punkts zu. Methoden werden abgerufen, indem man den Methodennamen und den Klammerinhalt verwendet:

```
v.info = 1;
v. showInfo ();
```

Man kann für eine Klasse eine beliebige Anzahl an Instanzen erstellen. Diese Instanzen sind unabhängig voneinander.

```
Cat v2 = new Cat();
v2. showInfo = 2;
v2. showInfo ();
```

Die instanziierten Methoden werden verwendet, um Operationen mit Objekten eines bestimmten Typs auszuführen. So lässt sich zB in der Klasse „My“ eine Methode deklarieren, welche den Wert des Attributs x um 1 erhöht:

```
class My {
  int x;
  void IncreaseA () {
        a++;
  }
}
```

Instanziierte Methoden (ebenso wie die Klassen-Methoden) können bestimmte Parameter und einen Rückgabewert haben. Sowohl die Parameter als auch der Rückgabewert werden in der Deklaration der Methode festgelegt.

```
class My {
        int x;
        // Adds dx to x and returns a new x value
        int moveX( int dx ) {
              x += dx;
              return x;
        }
}
```

10. Zugangsspezifikationssymbol

Zugangsspezifikationssymbole werden verwendet, um die Zugangsrechte zu individuellen Klassen, deren Operationen und Variablen festzulegen. Sie sind vor allem deshalb wichtig, weil sie Anwendungsdetails verstecken, damit sie ein Benutzer nicht sehen oder möglicherweise verwenden kann.

Öffentlich	Von jeder Klasse aus.
Privat	Nur innerhalb der vorgegebenen Klasse, kein Zugang von außen.
Geschützt	Von jeder Klasse desselben Pakets aus, oder ausgehend von irgendeinem Abkömmling der Klasse.
keines (paketfreundlich)	Von jeder Klasse desselben Pakets aus.

Als Beispiel wird hier ein Mitarbeiter mit einem bestimmten Alter und regulären Einkommen angeführt. Auch die IntroduceYourself-Methode kommt hier zum Einsatz.

```
class Employee {
    public Employee (int age, int wage) {
          this.age = age;
          this.wage = wage;
   }
   private int age = 1;
   public int getAge () { return age; }
   public void setAge(int age) { this. age = age; }
   private int wage = 1;

   public int getWage() { return wage; }
   public void setWage(int wage) { this.wage = wage; }
   public void introduceYourself(){
          System.out.println("My age a wage are "  + age +
          "years"+ wage + "Euros");
   }
   public static void main(String[] args) {
          Employee employee = new employee (30,100);
   }
}
```

11. Vererbung

Mittels Vererbung kann man eine Klassen-Hierarchie erstellen, wo sich jeder Knoten als ein spezieller Fall beliebiger Ahnenknoten definieren lässt. In der realen Welt würde man sagen, dass ein Stuhl eine Art von Möbelstück ist und ein Flugzeug ein Transportmittel. Ein Stuhl lässt sich jedoch nicht von Tieren unterscheiden, da diese auch vier Beine haben!
Um in Java einen Sub-Typ im Klassen-Header zu erstellen, und zwar direct nach dem Namen, wer-den das Schlüsselwort “extends” und der erweiterte Klassenname verwendet. Diese Klasse wird alle nicht-privaten Methoden (inklusive der paketfreundlichen Methoden, vorausgesetzt, die Er-weiterungsklasse befindet sich im selben Paket) und Ahnenvariablen der Klassen übernehmen, welche erneut deklariert und übereinander gelegt werden können.

Im Gegensatz dazu übernimmt die Klasse weder die privaten noch die statischen Methoden ihres Vorgängers, da sich diese nur auf eine bestimmte Klasse des Ahnen beziehen. Die finalen Methoden werden als Nachfahren bezeichnet und die Klasse übernimmt sie, kann sie jedoch nicht über-einander legen. Jedes Kindobjekt kann eine Typumwandlung oder ein Ahne sein.

11.1. Super

Werden Sub-Typ-Methoden abgerufen, stellt man oft fest, dass man nicht die gesamte Methode überlappen, sondern sie nur mit einer umfangreicheren Funktionalität ausstatten möchte. An diesem Punkt kann man eine *super.method ()* abrufen, womit man den Ahnen einer Funktionalität gleichsetzt. Man kann den Vorfahren auch einem Konstruktor gleichsetzen, indem man super () abruft. Dies muss allerdings der erste Abruf im Nachfahren-Konstruktor sein.

Um ein Beispiel zu geben, wird die Klasse Director erstellt, welche sich von der Klasse Employee ableitet. Dies bedeutet jedoch nicht, dass diese Klasse automatisch Zugriff auf alle privaten Variablen und Methoden der ursprünglichen Klasse haben soll (siehe Zugangsspezifikationssymbole).

Man setzt die Elternklasse einem Konstruktor gleich, indem man das Schlüsselwort “super” verwendet. Darüber hinaus muss es der erste Befehl im Körper des Konstruktors sein. Ruft man den Konstruktor der Elternklasse nicht ab, wird dieser automatisch in das Programm eingereiht – in diesem Fall als sogenannter impliziter Konstruktor, sprich ein Konstruktor ohne Parameter.

```
class Director extends Employee {
    public Director(int age, int wage)
    {
          super(age,wage);
    }
    public static void main(String[] args) {
          Employee k = new Director(30, 50);
    }
}
```

12. Polymorphismus

Man kann jede Methode in einer eigenen Kind-Implementation überlagern. Ruft man diese Methode in einem bestimmten Objekt ab, wird der überlappende Code immer ausgeführt – unabhängig davon, ob man sich der Methode mittels einer Referenz zum Vorfahrobjekt oder zu einem Nachfahrobjekt (dessen Klasse diese Überlappung enthält) nähert.

Diese Eigenschaft wird durch späte Bindung erreicht, wenn der Objekttyp, für welchen die Methode abgerufen wird, während der Laufzeit festgelegt wird, nicht während der Erstellung.

Dies lohnt sich jedoch nur für das Überlappen von Methoden, jedoch nicht für deren Überlastung. Hat man zwei Methoden für ein bestimmtes Objekt, welche sich nur durch die Parameter unter-scheiden (eine für den Ahnen, die andere für den Nachfahren), wird die Methode abgerufen, welche dieselben Parameter wie der aktuelle Verweis auf das Objekt hat. Der Abruf überlastender Methoden wird zum Zeitpunkt der Übersetzung festgelegt, wenn noch immer nicht klar sein muss, ob die Referenz auf einen Ahnen oder ein Nachfahrobjekt zeigt.

OBJEKTORIENTIERTE PROGRAMMIERUNG MIT JAVA

1. Klassen

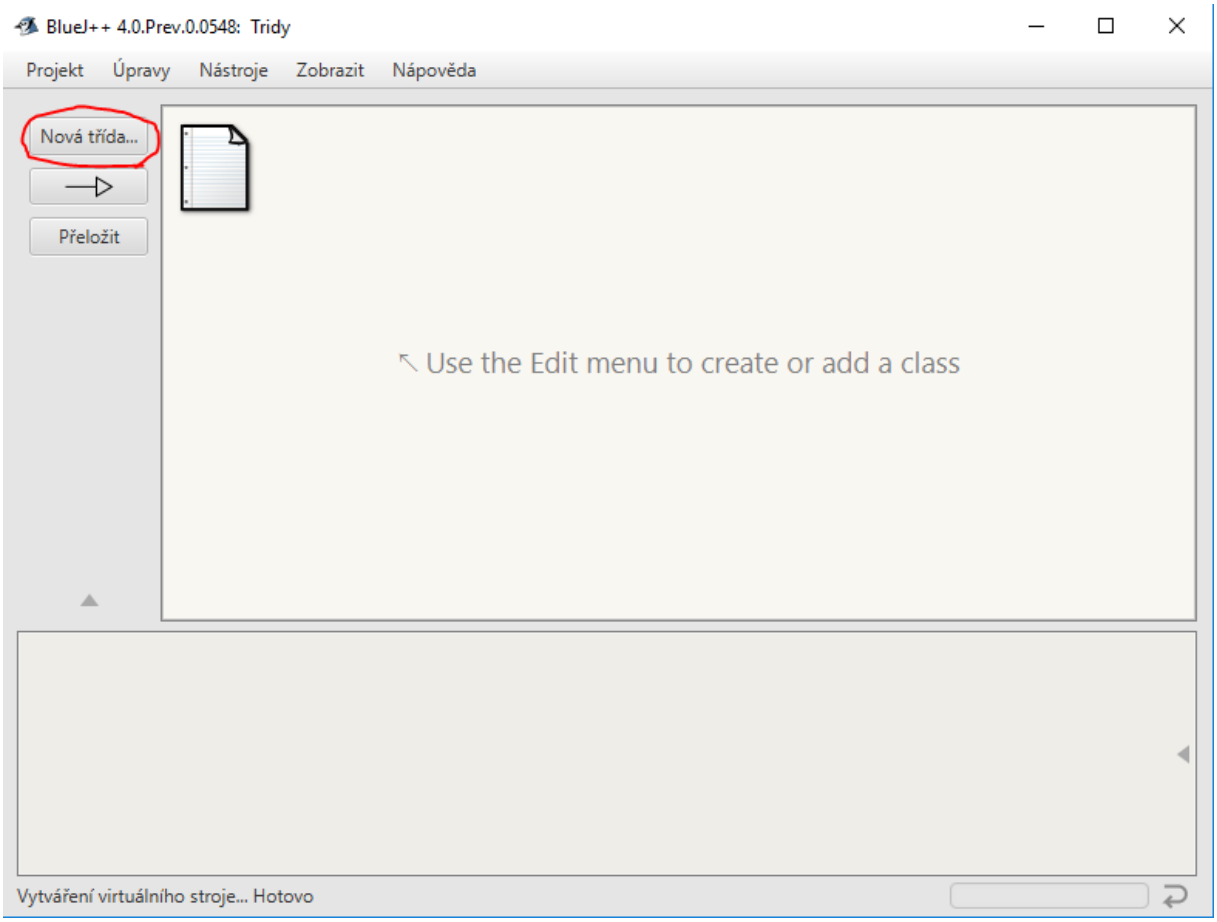
Im wirklichen Leben zum Beispiel wird das Wort Stuhl als Möbelstück bezeichnet. Die Funktion ist, darauf zu sitzen. Es handelt sich um eine Beschreibung des Objekts, die durch verschiedene Attribute oder Funktionen gekennzeichnet ist. Eine Analogie in der Programmierung ist eine Klasse. Die Klasse gruppiert Objekte mit einigen gemeinsamen Eigenschaften. Im wirklichen Leben gibt es viele verschiedene Stühle, die in Material und Farbe variieren können. Bei der Programmierung entspricht ein bestimmter Stuhl einer Instanz der entsprechenden Klasse, manchmal auch als Objekt bezeichnet. So kann die Klasse mit einem Formular verglichen werden, in das eine bestimmte Sache instanziiert wird. Typischerweise können Klassen beliebig viele Instanzen haben.

Einzelne Objekte können miteinander kommunizieren, sich gegenseitig verschiedene Nachrichten senden, in denen sie verschiedene Informationen oder Dienste anfordern können. Ein Beispiel kann ein Taschenrechner sein, den wir für viele Aufgaben benötigen, wie z.B. das Hinzufügen von zwei Zahlen. Jede der Funktionen dieses Rechners wird professionell als Methode bezeichnet. Das Anfordern der Verwendung einer bestimmten Methode wird als Methodenaufruf bezeichnet. Somit ist die Methode Teil des Programms, das eine Instanz als Antwort auf einen Methodenaufruf verwendet (eine Nachricht empfangen). Der Methoden-Builder definiert, wie das Objekt auf die Nachricht reagieren soll.

Das gesamte objektorientierte Programm Pecinovsky (2004) beschreibt als in einer Programmiersprache geschrieben eine Beschreibung der verwendeten Klassen, Objekte und Nachrichten, die diese Objekte senden, ergänzt durch komplexere Programme, indem es die Platzierung von Programmen auf einzelnen Computern beschreibt und diese der Verwaltung der jeweiligen Dienstprogramme zuordnet (z.B. Betriebssystem oder Anwendungsserver).

1.1. Erstellen von Klassen

Jetzt werden wir die Klassen selbst bauen. Also eine Art von Mustern, die von einer bestimmten Instanz (Objekt) erzeugt werden. Im BlueJ-Programm, das wir während dieses Kurses einsetzen werden, wählen wir ein neues Projekt aus. Wählen Sie dann die Schaltfläche auf der linken Seite, wie im Bild unten gezeigt.



Im sich öffnenden Fenster wählen wir die einfachste mögliche Definition einer Klasse - leere Klasse. Dann wählen wir den Namen der Klasse. In unserem Projekt haben wir uns für ErsteKlasse entschieden.

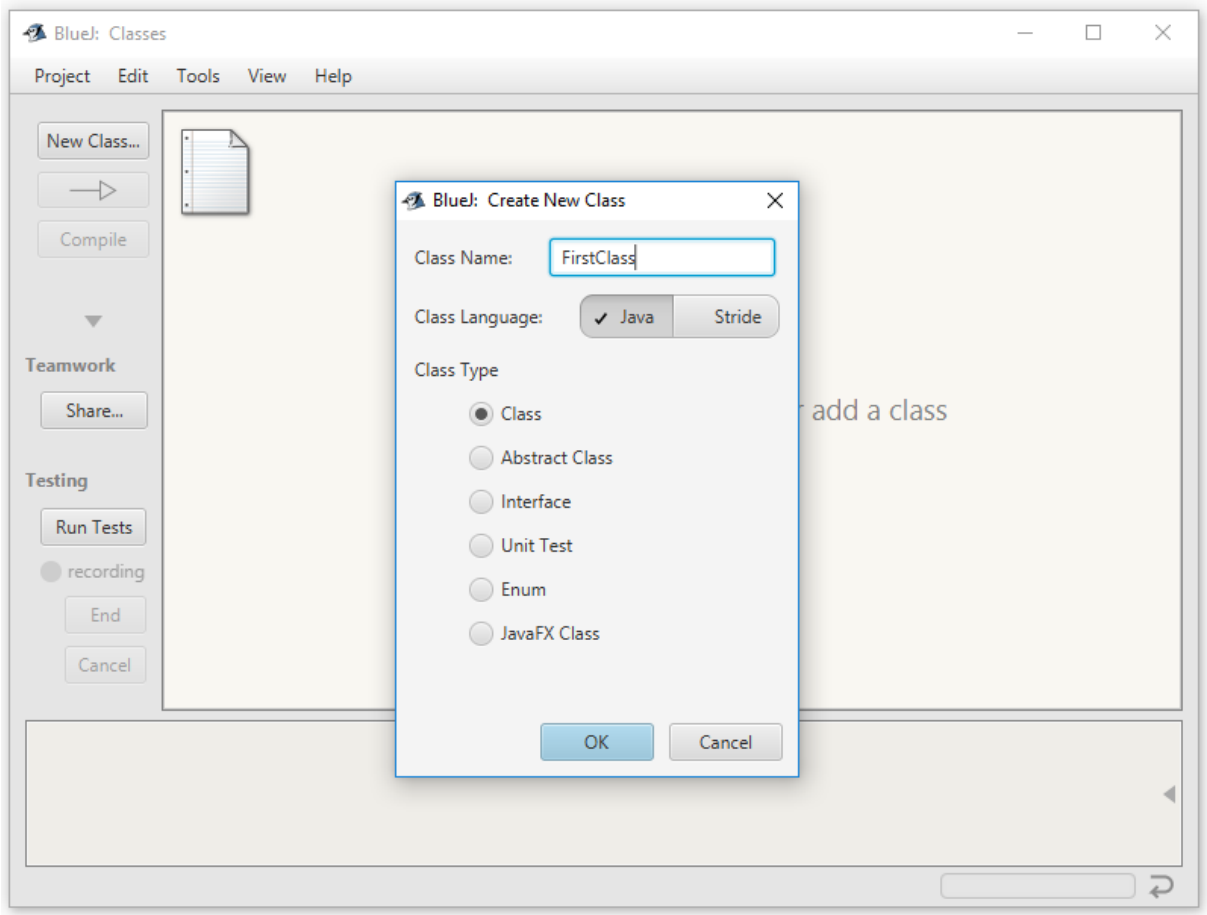
Der Name der Klasse hängt von einigen Faktoren ab:

Regeln für die Erstellung eines Identifikators

Links, Klassen und andere müssen für die korrekte Identifizierung benannt werden. Diesen Namen nennen wir Identifikatoren. Diese Identifikatoren müssen eine Reihe von Regeln erfüllen:

- Es kann alle Zeichen enthalten, die in einem Satz von UNICODE enthalten sind.
- Lücken dürfen nicht verwendet werden.
- Es ist üblich, mehrwertige Klassennamen ohne Leerzeichen zu schreiben. Einzelne Wörter beginnen dann mit Großbuchstaben wie: MeineErsteKlasse
- Groß- und Kleinschreibung werden als unterschiedlich angesehen.
- Die Länge ist unbegrenzt

- Es darf nicht mit einer Ziffer beginnen.
- Kann nicht mit den Keywords übereinstimmen



Nach der Erstellung und Übersetzung haben wir die erste Klasse. Wenn wir in den Ordner schauen, in dem wir das gesamte Projekt gespeichert haben, werden wir feststellen, dass sich mehrere Dateien im Ordner befinden.

FirstClass.java: Wir nennen diese Datei Quelldatei. Wir werden ein Programm schreiben, welches das Verhalten der Klasse und damit ihre Instanzen beschreibt. Diese Datei ist nur eine Textdatei. Sie können es in jedem beliebigen Texteditor bearbeiten.

FirstClass.class: Ein übersetzter Bytecode wird in dieser Datei gespeichert.

FirstClass.ctxt BlueJ: zusätzliche Datei. Wenn Sie diese Datei löschen, wird BlueJ sie bei der nächsten Übersetzung erneut erstellen.

1.2. Arbeiten mit der Klasse

Nach einem Doppelklick auf die Klasse öffnen sich ein Textfenster, in dem Sie den Quellcode der Klasse eingeben oder bearbeiten können. Siehe Bild unten:

Im Textfenster werden wir sehen:

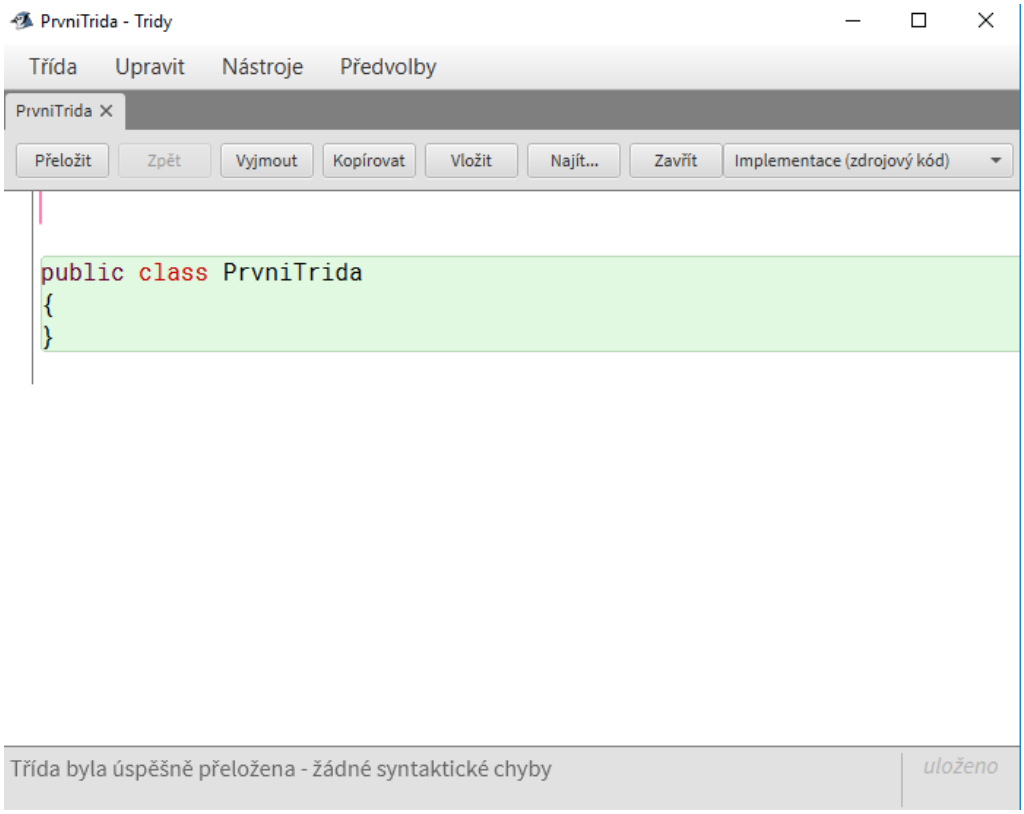
- Klassendefinition FirstClass
- Konstrukteur FirstClass ()
- Methode sampleMethod()

Die Klassendefinition enthält Wörter

Public: Ein Schlüsselwort, das angibt, dass jeder mit dieser Klasse arbeiten kann.

Class Ein Schlüsselwort, das die Klassendefinition angibt.

FirstClass: Klassenname (sein Identifikator)
Der Inhalt der Klasse ist in den folgenden Klammern enthalten. In unserem Beispiel enthält sie noch keine Informationen.



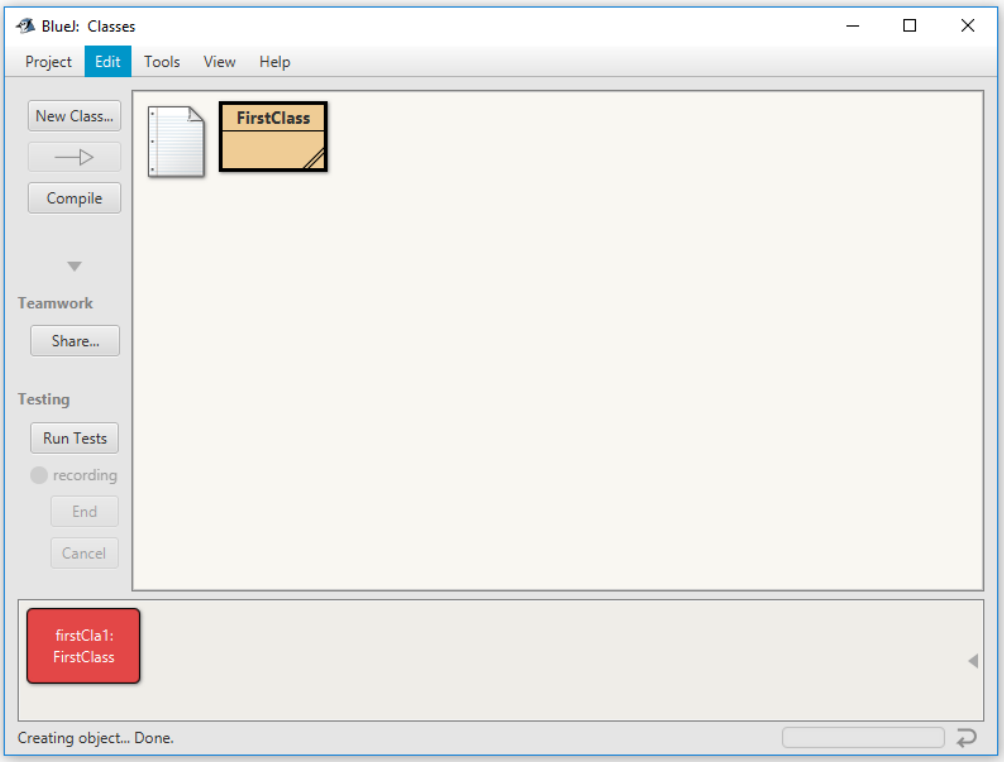
1.3. Erstellen einer ersten Instanz

Schließen Sie das Textfenster. Wenn Sie mit der rechten Maustaste auf unsere erste Klasse klicken. Achtung! Die betreffende Klasse muss kompiliert werden! Wählen Sie im Popup-Menü den neuen Befehl ErsteKlasse (). BlueJ wird uns dann nach dem Namen der zu erstellenden Instanz fragen. Gleichzeitig werden wir es vor dem ausgewählten Namen anbieten. Alles, was du tun musst, ist zu genehmigen. So haben wir die erste Klasse erstellt. Wie im Bild unten gezeigt.

Durch Auswählen des neuen Befehls ErsteKlasse () erstellen wir eine Instanz der Klasse, indem wir eine Nachricht senden, die aus dem neuen Schlüsselwort kompiliert wurde, gefolgt vom Namen der Klasse, aus der wir ein Paar runde Klammern erstellen möchten. Also rufen wir eine spezielle Methode namens Konstruktor auf. Der Konstruktor erstellt die angeforderte Instanz und gibt einen Link zurück, über den wir die erstellte Instanz ansprechen.

In unserem Fall haben wir keine Konstruktormethode konstruiert. Aber der Konstruktor muss jede Klasse haben! Wenn wir keinen Konstruktor erstellen, erstellt der Compiler automatisch den einfachsten Konstruktor, den wir bestimmen - den Standardkonstruktor. Im Moment erstellen wir den ersten Konstruktor in der Klasse, der Compiler stoppt mit der Erstellung des impliziten Konstruktors.

Die erzeugte Instanz der Klasse ist in der folgenden Abbildung dargestellt. Es ist unten, in dem Bereich, den wir die Objektbank nennen. Die Instanz der Klasse ist rot.



1.4. Eine Klasse entfernen

Sehr einfach. Klicken Sie mit der rechten Maustaste auf die Klasse und wählen Sie Löschen. Nach der Bestätigung wird die Klasse tatsächlich gelöscht. Die Instanz der Klasse selbst blieb jedoch erhalten. Die Anforderung, eine Klasse zu löschen, ist eigentlich eine Anforderung, die entsprechenden Dateien von der Festplatte zu löschen. Die Klasseninstanz selbst wird im Speicher abgelegt. Wenn wir die Instanz der Klasse entfernen wollen, müssen wir die virtuelle Maschine neu starten.

1.5. Neustart der virtuellen Maschine

Beim Neustart der virtuellen Maschine werden alle Referenzen im Link-Stapel gelöscht. Also löschen wir alle Instanzen. Wir können dies entweder mit der Tastenkombination Strg + Umschalt + r oder durch einen Rechtsklick auf das Rechteck unten rechts und die Auswahl des Befehls tun: Starten Sie die virtuelle Maschine neu.

2. Konstrukteure

Im vorherigen Kapitel hat der Compiler einen impliziten Konstruktor für uns erstellt. Nun zeigen wir Ihnen, wie Sie Ihre eigenen Konstruktoren erstellen können.

2.1. Nichtparametrischer Konstruktor

Erstellen Sie eine Klasse “Student” ähnlich derjenigen, die wir gelöscht haben. Jetzt erstellen wir einen nicht-parametrischen Konstruktor. Nichtparametrisch bedeutet, dass es für seinen Lauf keine Informationen - Parameter - benötigt. Die Klassen- und Konstruktordeklaration selbst könnte in etwa so aussehen. Der Konstruktor selbst ist gelb:
public class Student

```
{
        int math =3;
              int english =3;
              int ICT =3;
        public Student ()
        {
              .........
        }
}
```

Nun gehen wir die Bedeutung der einzelnen Textteile durch.

Wir haben zuerst die Klassen Student und Variable Mathematik, Englisch und ICT erklärt.

Der Header des Konstruktors sieht dem Klassenkopf sehr ähnlich. Das Keyword class wird weggelassen. Der Name des Konstruktors muss mit dem Klassennamen übereinstimmen, in dem der Konstruktor aufgeführt ist. Hier sind die Klammern, in die die Parameter geschrieben werden, dann der parametrische Konstruktor, wenn keine Klammern geschrieben werden, wird es ein parametrischer Konstruktor sein. Der Körper des Konstruktors selbst steht in Klammern.

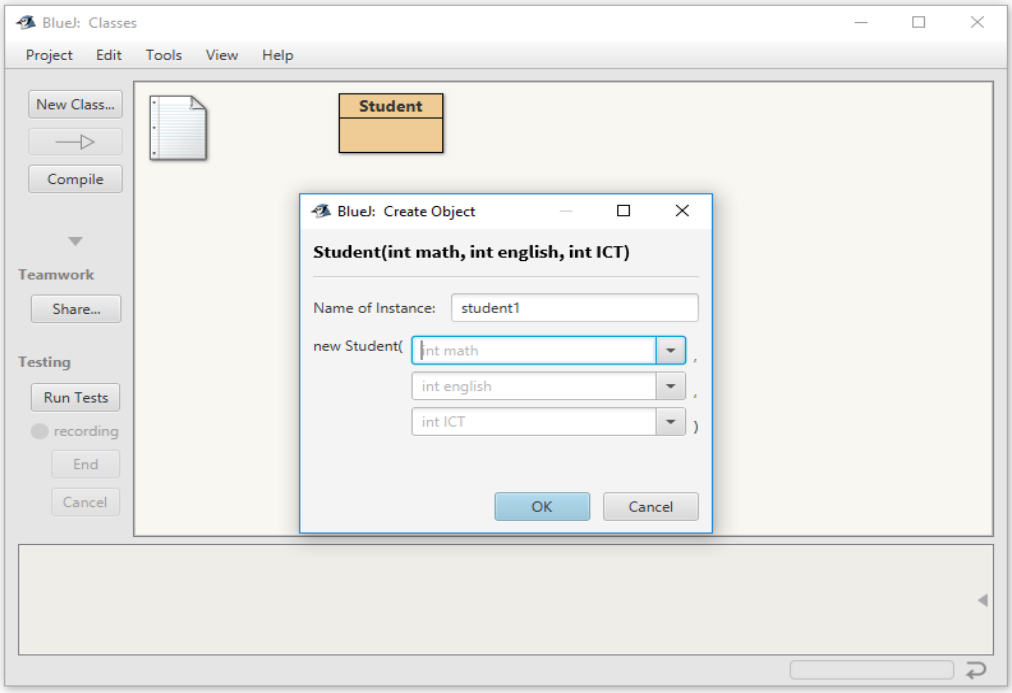
2.2. Konstruktor mit Parametern

Wenn wir eine Studenteninstanz wären, hätte jeder Student die gleichen Noten. Deshalb erstellen wir diesmal einen anderen Konstruktor. Diesmal mit den Parametern.

```
public class Student
{
      int math =3;
      int english =3;
      int ICT =3;
      public Student ()
      {
            .........
      }
      public Student (int math,int english,int ICT)
      {
            .........
      }
}
```

Der neue parametrische Konstruktor wurde gelb markiert. Beide Konstruktoren haben den gleichen Namen. Der Übersetzer unterscheidet sie nach Anzahl und Art der Parameter. Dem parametrischen Konstruktor geben wir zunächst den Typ und damit die Kennung an, mit der das Programm den Parameter im Körper des Konstruktors aufruft. Daher können wir keine Konstruktoren mit gleichen Parametertypen konstruieren. Der Übersetzer unterscheidet nicht einmal die Rückgabewerte für einzelne Konstruktoren. Die Verwendung mehrerer Konstruktoren wird als Überladung bezeichnet.

Nun, da wir eine neue Instanz erstellen wollen - einen Schüler mit einem parametrischen Konstruktor -, werden wir aufgefordert, ganzzahlige Parameter einzugeben, wie in der folgenden Abbildung gezeigt.



2.3. This

Viele Konstrukteure sind einander ähnlich. Häufig wollen wir in einem Konstruktor einen anderen Konstruktor verwenden. Dieses Umschreiben des Body-Konstruktors kann durch die Verwendung des "This" Schlüsselwortes vermieden werden, gefolgt von einer Liste von Parametern. Hüten Sie sich davor, einen anderen Konstruktor aufzurufen, indem Sie This verwenden. This muss der allererste Konstruktorbefehl sein! Die Verwendung des This Konstruktors ist ein weiteres Beispiel.

```
public class Student
{
      int math =3;
      int english =3;
      int ICT =3;
      public Student ()
      {
            this (0, 0, 0);
      }
      public Student (int math,int english,int ICT)
      {
            .........
      }
}
```

3. Methoden

Die Methode ist ein spezifisches Unterprogramm, das eine bestimmte Funktion ausführt. Auch ist es eines der am häufigsten verwendeten Werkzeuge (fast) jeder Programmiersprache. Wir könnten die Methoden mit den Werkzeugen vergleichen, mit denen dann jede Instanz der Klasse ausgestattet ist. Das Verfahren selbst besteht aus mehreren Teilen:

- Zugriffsspezifikation - die bestimmt, wer die Methode aufrufen darf. Die häufigsten sind öffentlich und privat. Der Spezifikator ist optional.
- Art des Rückgabewertes - Obligatorisch. Wenn die Methode nicht zurückkehrt, schreiben wir void.
- Methodenname - die gleichen Regeln wie in Konstruktoren
- Methodenparameterliste - auch das gleiche Prinzip wie bei den Konstruktoren

Der Körper des Verfahrens selbst ist in Verbundklammern eingeschlossen. Wo wir einzelne Befehle schreiben können. Wenn wir wollen, müssen wir nichts in den Körper schreiben.

```
public void Hello()
{
      System.out.println(”Hello”);
}
```

3.1. Methoden, die einen Wert zurückgeben

Wenn die Methode einen Wert zurückgeben soll, müssen wir ihren Typ in der Kopfzeile angeben, und im Body der Methode müssen wir die Anweisung zurückgeben, gefolgt von der Variablen, die wir zurückgeben möchten. Nach der Return-Anweisung wird die Methode sofort beendet. Daher wird der Code, der auf die Return-Anweisung folgt, nicht im Body der Methode ausgeführt. Das Beispiel ist unten dargestellt.

```
public double averageGrade ()
{
      double average= (math + english + ICT)/3;
      return average;
}
```

3.2. Aufruf von Methoden

Durch einfaches Schreiben von Codes wird kein Code ausgeführt. Es wird geschehen, wenn wir es bestimmen. Sie können die Methode nur an der Stelle aufrufen, an der die Methode zugänglich ist. Rufen Sie die Methode durch Eingabe des Methodennamens auf und geben Sie die Parameter der Methode in Klammern ein. Wenn die Methode in einer anderen Klasse aufgeführt ist, müssen wir zuerst die Klasse benennen. Der Methodenname wird durch einen Punkt getrennt. Wenn wir eine Instanznachricht senden, müssen wir zuerst eine Referenz auf diese Instanz schreiben. Bei Attributen ist die Situation ähnlich.

3.3. Parameterübergabe an Methoden

Werte primitiver Typen wie Zeichen, logische Werte oder Zahlen werden übergeben, so dass der Wert in die lokale Methodenvariable kopiert wird.

Objekttypwerte werden über die Verknüpfung übergeben. Daher ein Link auf das Objekt, in dem der Aktualparameter gerade in die lokale Methodenvariable kopiert wird.

4. Statische Attribute

Statische Elemente gehören zu einer Klasse und nicht zu einer Instanz. Statische Attribute werden mit “static” bezeichnet. Da es zu einer Klasse gehört, haben alle Methoden dieser Klasse Zugriff darauf. Wir können statische Attribute lesen, auch wenn es keine Klasseninstanz gibt. Die Deklaration der statischen Attribute ist unten aufgeführt und gelb markiert.

```
class Group {
      private static int number = 15;
      public void New(int count) {
            number = number + count;
      }
}
```

4.1. Statische Klassen

Klassenmethoden werden in der Klasse aufgerufen. Dies sind die Hilfsmethoden, die wir oft verwenden, aber wir wollen keine Instanz speziell für diesen Zweck erstellen. Als Beispiel kann eine statische Methode verwendet werden, um zu testen, ob eine bestimmte Zahl ein Plus ist.

```
public static boolean isPlus(int number) {
      if (number >= 0) {
            return true;
      }
      return false;
}
```

Achtung! Da die statische Methode zur Klasse gehört, können wir nicht auf Instanzattribute in ihr zugreifen. Diese Attribute existieren nicht innerhalb einer Klasse, sondern innerhalb einer Instanz.

4.2. Lokale Variablen

Manchmal müssen wir uns an etwas in der Methode erinnern. Zu diesem Zweck werden lokale Variablen verwendet. Wir deklarieren sie innerhalb der Methode. Über die Methode hinaus kann nicht auf sie zugegriffen werden. Dies ermöglicht es uns, eine weitere lokale Variable mit dem gleichen Namen in einer anderen Methode zu definieren. Wir verwenden in ihrer Deklaration keine Zugriffsmodifikatoren (z.B. öffentlich und privat) oder static. Wir müssen ihrer Deklaration einen gewissen Wert beimessen. Beim Verlassen der Methode wird die lokale Variable gelöscht. Deshalb gibt es in ihnen nichts, was wir zwischen den verschiedenen Methoden brauchen. Häufige Gründe für die Verwendung lokaler Variablen sind, das Programm transparenter zu machen und die Anzahl der Fehler zu reduzieren, die durch die wiederholte Typisierung komplexer Ausdrücke verursacht werden. Die Deklaration der lokalen Variablen erfolgt wie im folgenden Beispiel:

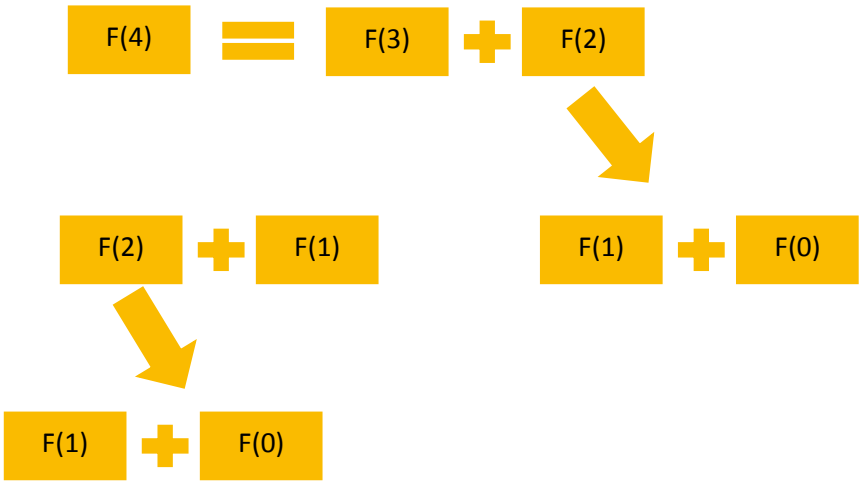
```
public void totalPrice (int pieces)
{
      int totalPrice = pieces *15;
}
```

4.3. Rekursion

Wiederkehrend ist die Definition eines Objekts (mathematisch verstanden) durch dich selbst. Rekursive Merkmale müssen einen Mechanismus beinhalten, der die Rekursion zu gegebener Zeit beendet. Wenn das nicht geschehen würde, würde die Rekursion bis zur "Unendlichkeit" gehen. Das klassische Beispiel für das Ende der Rekursion ist das Einfügen von Stopps.

Ein klassisches Beispiel für Rekursion ist die Fibonacci-Sequenz. Bei der Fibonacci-Sequenz ist jedes Element der Sequenz die Summe seiner beiden vorherigen Elemente. A F (0) = 0 und F (1) = 1.

Ein Beispiel für die Berechnung des vierten Elements ist in der folgenden Abbildung dargestellt. Aus der Abbildung ist ersichtlich, dass wir, um F (4) zu berechnen, zunächst F (3) und F (2) rekursiv neu berechnen müssen. Um F (3) zu berechnen, müssen wir zuerst F (2) + F (1) berechnen, und für F (2) müssen wir F (1) + F (0) rekursiv wiederholen.



Ein großer Teil der Berechnung und Rekursion erfolgt wiederholt, was sie rechenintensiv macht. Aus diesem Grund ist es oft besser, klassische Zyklen zu verwenden. In einigen Fällen wird der gesamte Algorithmus rekursiv definiert (z.B. Fibonacci-Sequenz), oder die Rekursion kann die Arbeit mit einigen Datenstrukturen erleichtern.

Ein konkretes Beispiel für einen rekursiven Funktionsaufruf ist unten dargestellt. In unserem Beispiel haben wir 2 gelb markierte Stationen. Der rekursive Aufruf ist dann mit einer grünen Farbe markiert.

```
public static int fib (int n){
     if(n == 0) return 0;
     else if(n == 1) return 1;
      else return fib (n - 1) + fib (n - 2);
}
```

5. Verkapselung

Wikipedia (vom 7. Mai 2018) definiert die Verkapselung:

Die Verkapselung ist eine der Grundlagen der OOP (Objektorientierte Programmierung). Es bezieht sich auf die Bündelung von Daten mit den Methoden, die mit diesen Daten arbeiten.[5] Die Kapselung wird verwendet, um die Werte oder den Zustand eines strukturierten Datenobjekts innerhalb einer Klasse zu verbergen und den direkten Zugriff Unbefugter auf diese zu verhindern. Öffentlich zugängliche Methoden werden in der Regel in der Klasse (sogenannte Getter und Setter) für den Zugriff auf die Werte bereitgestellt, und andere Client-Klassen rufen diese Methoden auf, um die Werte innerhalb des Objekts abzurufen und zu modifizieren.

Die Verkapselung ist sehr wichtig. Wenn wir ein kompliziertes Programm hätten, könnten wir versehentlich den Verlauf oder einige seiner Teile beeinflussen. Das Auffinden und Beheben solcher Probleme würde uns viel Zeit kosten. Die Verkapselung ist eines der grundlegenden Konzepte der Objektprogrammierung.

Die Verkapselung in Java ist ein Mechanismus zum Packen von Daten (Variablen) und Code. In der Verkapselung werden die Klassenvariablen vor anderen Klassen verborgen und können nur mit Methoden ihrer aktuellen Klasse angesprochen werden. Daher wird es auch als Ausblenden von Daten bezeichnet.

Dies erreichen wir, indem wir die Teile, auf die andere Funktionen zugreifen sollen, als öffentlich auswählen. Dieser öffentliche Teil der Klasse wird als Klassenschnittstelle bezeichnet. In der Klassenschnittstelle ist es nur sinnvoll, das aufzunehmen, was die anderen Abschnitte des Programms wirklich über die Klasse wissen müssen. Für alle anderen wollen wir die anderen Teile des Programms nicht verwenden und setzen sie privat.

Wenn einige Teile des Programms einige Abschnitte der öffentlichen Klasse verwenden, die später geändert werden, kann dies die Funktionalität beeinträchtigen. Daher ist es besser, seine öffentlich zugänglichen Teile nach der Veröffentlichung der Klasse nicht zu ändern.

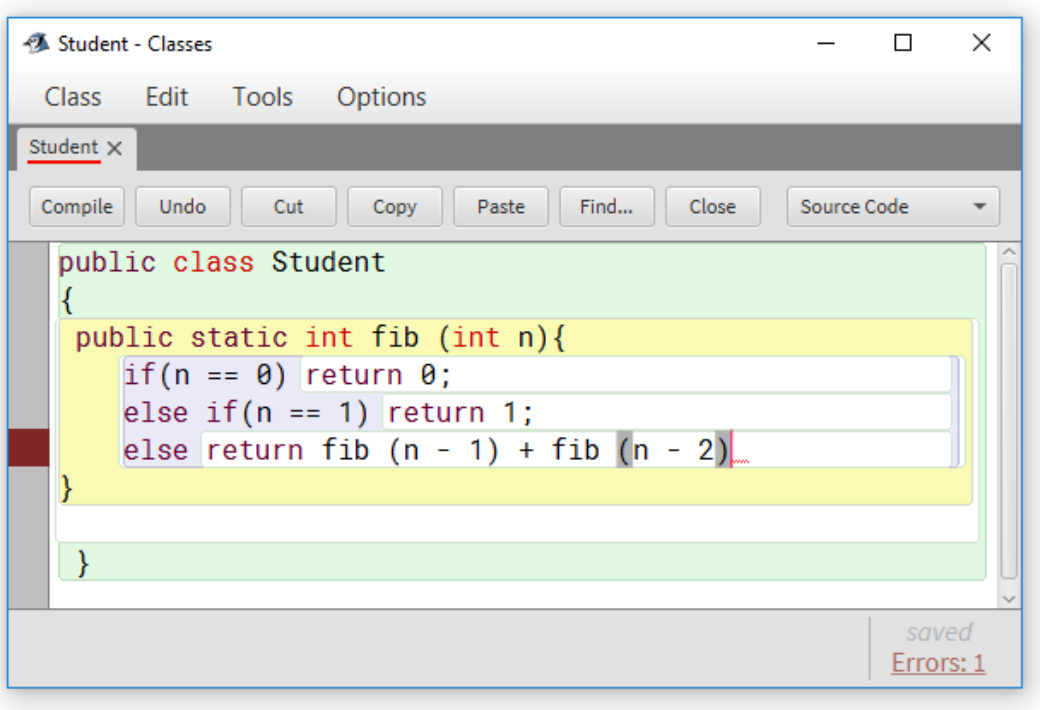
Häufig werden alle Klassenattribute als privat deklariert. Nach unserem Ermessen können wir Zugriffsmethoden (Setter, Getter) veröffentlichen, mit denen sichergestellt werden kann, dass das betreffende Attribut nur gelesen und nicht bearbeitet werden kann oder mit einigen einschränkenden Bedingungen gesetzt werden kann. (Zum Beispiel ist das Alter nur eine positive ganze Zahl). Wenn der Attributname mit dem Namen des Parameters übereinstimmt und ich mit beiden arbeiten muss, verwende ich das Schlüsselwort this. Dieses Wort wird als Klassenname verwendet, in dem die Methode enthalten ist.

6. Debuggen des Programms

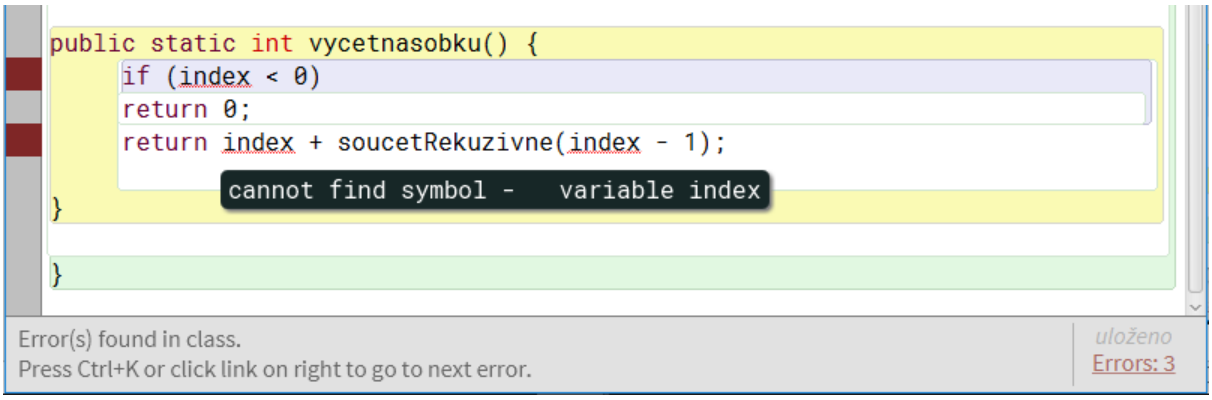
Wir machen oft einen Fehler, wenn wir ein Programm schreiben. Diese Fehler können in mehrere Kategorien eingeteilt werden.

Häufige Probleme sind syntaktische Fehler. Wann überwinden wir die Sprachsyntax? Also gegen die Regeln, wie man einzelne Teile zusammensetzt. Typischerweise wird z.B. ein vergessenes Semikolon oder eine Klammer verwendet.

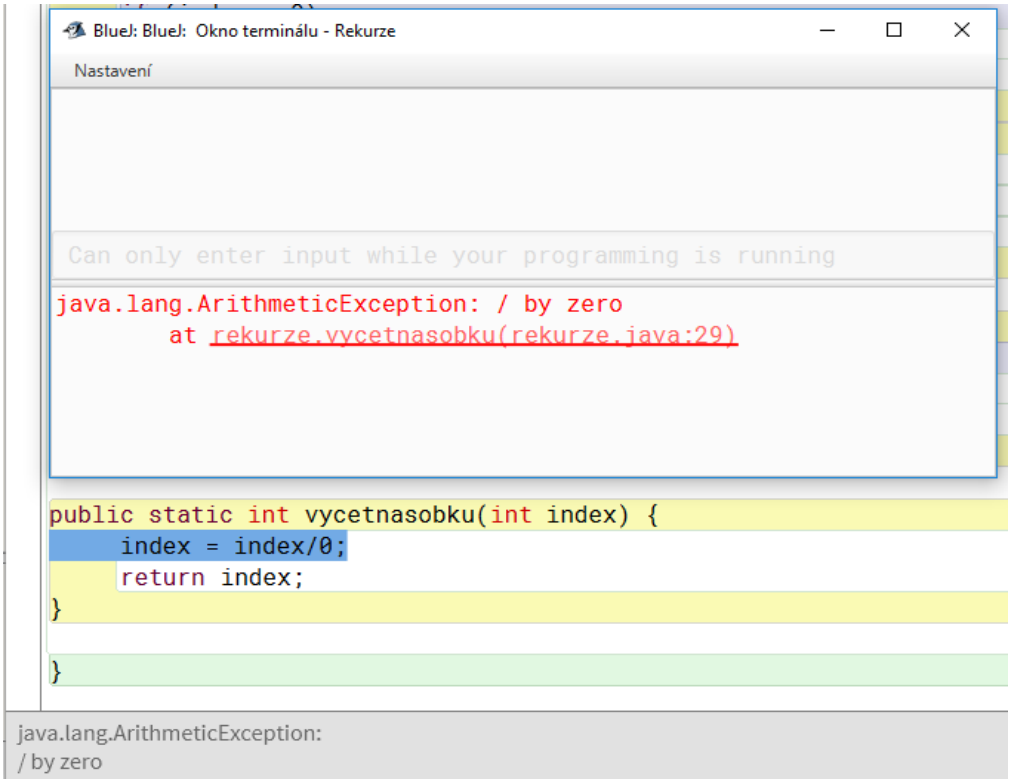
Dieses Problem wird sehr oft durch den Compiler vor der Kompilierung selbst gekennzeichnet. Gleichzeitig werden wir die Linie, in der sie das Problem annimmt, bunt markieren. Wie im Bild zu sehen:



Andere Fehler treten beim Kompilieren auf. Für weitere Informationen über den Fehler können wir unten links zu dem Wort Errors springen. Nach dem Anklicken erhalten wir weitere Hilfe zum Fehler.



Andere Fehler können als Laufzeitfehler bezeichnet werden. Diese Fehler werden vom Compiler nicht gefunden. Aber in einigen Phasen des Programmlaufs können sie auftreten. Ein typisches Beispiel ist die Nullteilung. Wenn das Programm stoppt, wenn der Nullbruch eintritt, öffnet sich das Terminalfenster, in dem es als Fehler angezeigt wird. BlueJ markiert auch die Zeile, in der das Problem aufgetreten ist, wie im Bild unten gezeigt. Um solche Fehler zu beseitigen ist es notwendig, alle möglichen potentiell problematischen Zustände des Programms im Idealfall auszuprobieren. So, dass der Benutzer nicht auf ähnliche Probleme stößt. Im Idealfall bereiten wir im Vorfeld eine Reihe von Testaufgaben vor.



Semantische Fehler sind die heimtückischste Art von Fehlern. Wenn der Compiler nicht erscheint, scheint das Programm nahtlos zu funktionieren. In einigen unerwarteten

Momenten zeigt das Programm jedoch einen Fehler an. Was ein großes Problem sein kann.

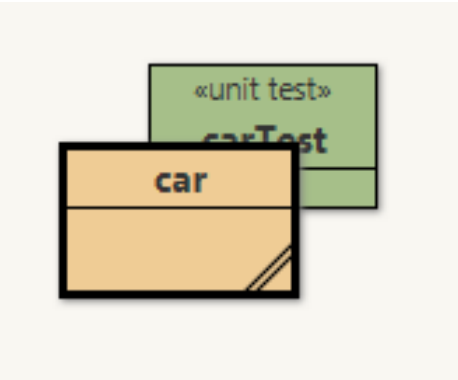
Ein Beispiel könnte sein: Mars Landing Marine Crash (1999) Problem der Kommunikation zwischen den Komponenten - der Benutzer der Schnittstelle erwartete den Wert in Kilometern, der Anbieter gab ihn in Meilen an. Statt der geplanten 140-150 Kilometer führte er nur 57 Kilometer über die Oberfläche. In dieser Höhe ist die Marsatmosphäre auf der Sonde jedoch zu dicht. Der Climate Orbiter brannte auf etwa 80 Kilometern. (Quelle: Technet.cz)

Diese Art von Fehler wird durch Software-Engineering gelöst.

6.1. Testgetriebene Entwicklung

Zusätzlich zum Testen des fertigen Programms können wir eine andere Philosophie der Softwareentwicklung wählen - Test Driven Development. Als Teil dieses Ansatzes definieren wir zunächst eine Reihe von Tests und schreiben dann das Programm selbst, in dem wir uns nur darauf konzentrieren, den Code durch diese Tests zu leiten. (Wir sprechen z.B. nicht von Code-Effizienz) Das Refactoring folgt. Duplikate werden aus dem Code entfernt, und der Code wird in der Regel in der akzeptabelsten Form bearbeitet. Durch die Wiederholung von Tests wird sichergestellt, dass die Funktionalität des Codes während des Refactorings nicht beeinträchtigt wird.

BlueJ bietet zu diesem Zweck eine Reihe von Tools an. In BlueJ erstellen wir einen Komponententest für diese Klasse, indem wir mit der rechten Maustaste auf die zu testende Klasse klicken und die Option zum Erstellen einer Testklasse auswählen. Die gegebene Klasse erhält dann einen untrennbaren Partner - einen Komponententest. Die Testklasse wird farblich differenziert. Wie im Bild zu sehen.



Wenn wir die Tests mit der gleichen Menge von Objekten durchführen wollen, können wir einen Komponententest erstellen. Wir erstellen das Tool, indem wir nur die Objekte erstellen, die sie im Testwerkzeug im Objektstapel enthalten sollen. Klicken Sie nun mit

der rechten Maustaste auf die Testklasse und wählen Sie Object Bench to Test Fixture. Beachten Sie, dass alle Nachrichten, die wir seit dem letzten Neustart der virtuellen Maschine gesendet haben, aufgezeichnet wurden. Wenn wir also sicher sein wollen, was getan wird, starten Sie zuerst die virtuelle Maschine neu und erstellen Sie dann Objekte. Wenn wir ein bereits gespeichertes Objekt mit anderen Objekten überschreiben wollen, erstellen wir einfach diese Objekte und wählen dann Object Bench to Test Fixture. Das Produkt wird überschrieben.

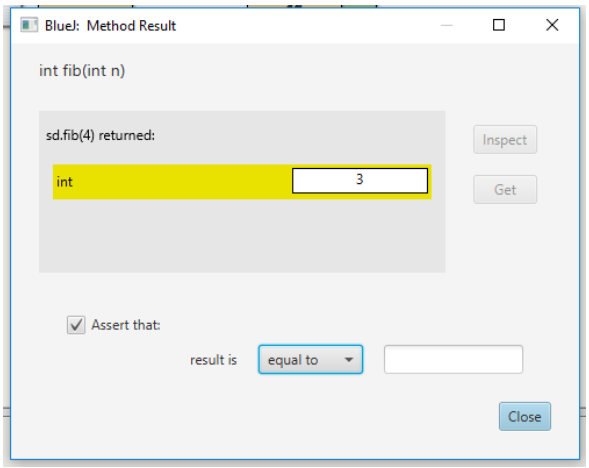
Alternativ können wir die Testklasse auch manuell bearbeiten.

Um anschließend Objekte aufzurufen, klicken Sie mit der rechten Maustaste auf die Testklasse und wählen Sie Test Fixture to Object Bench.

6.2. Einen Test erstellen

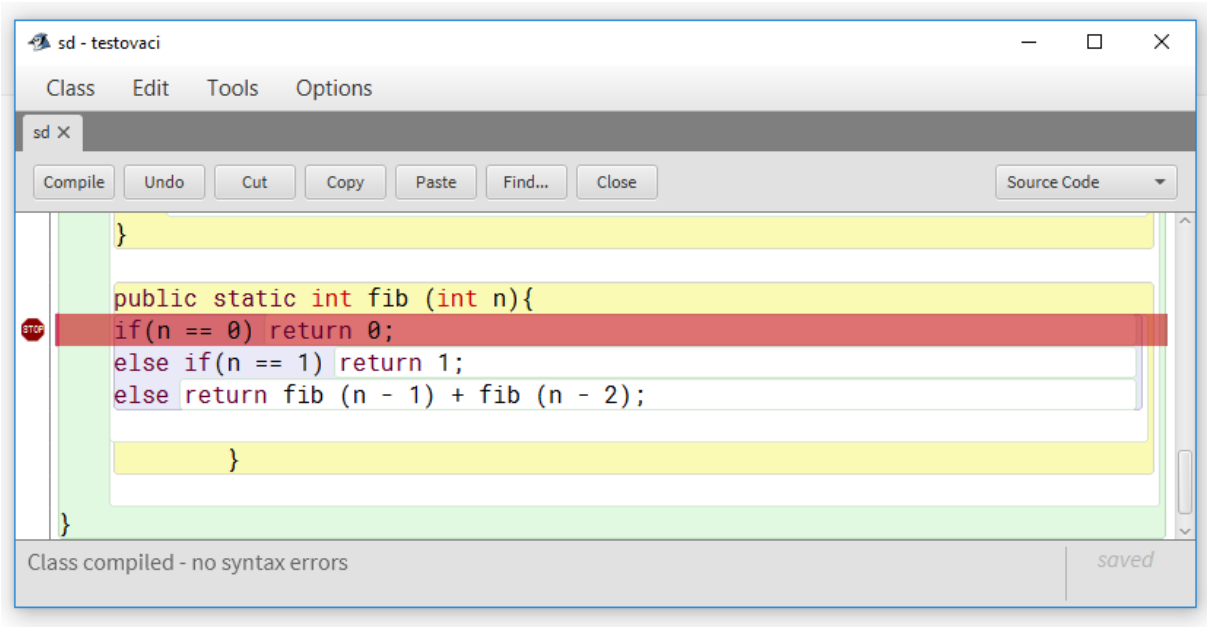
Im Idealfall starten wir zuerst die virtuelle Maschine neu. Klicken Sie nun mit der rechten Maustaste auf den Komponententest und wählen Sie Testmethode erstellen. Wählen Sie zunächst den Namen der Testmethode selbst. Jetzt zeichnet BlueJ unser Verfahren zur Prüfung des Produkts auf. Wir werden die erforderlichen Maßnahmen ergreifen. Während der Aufzeichnung wird BlueJ uns bitten, den Rückgabewert für ausgewählte Methoden zu testen. Es ist auf dem Bild unten dargestellt. Während der Aufnahme leuchtet das rote Licht links. Wenn wir die Aufnahme stoppen wollen, ohne zu speichern, wählen wir: Abbrechen. Um zu beenden und zu speichern, wählen wir den Ausgang. Beide befinden sich direkt unter dem roten Knopf.

Wenn wir den Test testen wollen, klicken Sie mit der rechten Maustaste auf den Komponententest, wo wir den Test im Menü auswählen. Wenn wir im Laufe des Tests anders reagieren als der Rückgabewert des Tests, wird der Test unterbrochen. Es öffnet sich ein Testergebnisfenster, in dem wir erfahren können, in welchem Teil des Codes der Fehler aufgetreten ist.



7. Debugger

Es ist ein Werkzeug, das dem Programmierer hilft, Fehler im Programm zu erkennen. Um den Debugger wiederherzustellen, müssen wir zuerst einen Stopp im Code setzen. Was wir tun werden, indem wir im Code-Editor auf die linke Spalte klicken. Auf der entsprechenden Zeile. Hier erscheint die rote Stoppmarke und die Linie ist rot markiert. Wie im Bild unten gezeigt. Wenn das Programm ausgeführt wird, stoppt das Programm an der Stoppstelle und es erscheint ein Debuggerfenster.



Am unteren Rand des Debugger-Fensters (unten) befinden sich insgesamt 5 Schaltflächen.

- **stopp**, um das Programm zu stoppen. Es ist z.B. nützlich, wenn sich das Programm aufgehängt hat.

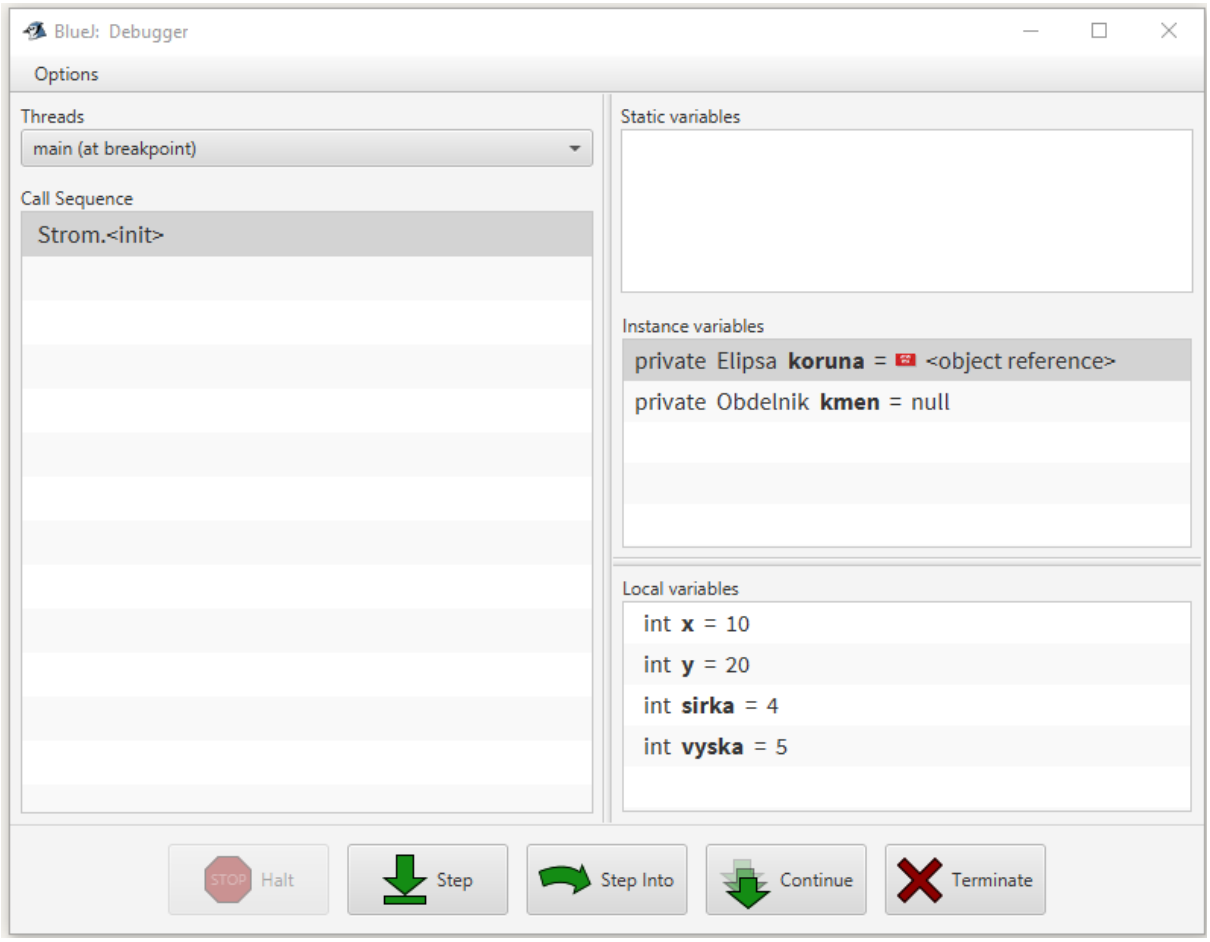
- **step**, um den nächsten Schritt des Codes auszuführen. Der jeweilige Schritt wird auch grafisch durch die grüne Farbe im Code-Editor angezeigt.

- **step into** ist dem step sehr ähnlich. Der Unterschied besteht darin, dass, wenn Sie die step aufrufen, die gesamte Methode ausgeführt wird, aber wenn Sie step into wählen, die aufgerufenen Methode Schritt für Schritt ausgeführt wird.

- **continue**, das Programm wird normalerweise bis zu seinem Ende oder bis zum nächsten Halt fortgesetzt.

- **terminate**, um die Programmausführung zu beenden. Alternativ können wir das Programm beenden, indem wir die virtuelle Maschine neu starten.

Wir können diese neuen Stopps auch während des Prozesses dem Code hinzufügen. Das Debugger-Fenster selbst ist in mehrere Teile unterteilt. Oben links können wir einen bestimmten Thread auswählen. Unter der Wahl des Threads ist der Bereich Calling Order. Dieser Bereich könnte als Rückadress-Stapel bezeichnet werden. Hier sind alle Aufrufe während des Programmlaufs aufgelistet. Neue Anrufe werden am höchsten eingestuft. Wenn wir auf einen bestimmten Aufruf klicken, öffnet sich der Code-Editor mit der Klasse, die die aufgerufene Methode enthält. Die Farbe wird durch die aktuell ausgeführte Zeile hervorgehoben.



Im rechten Teil des Debuggerfensters werden die Klassenattribute aufgelistet. Dies sind Klassenattribute, die zu dem aktuell ausgewählten Ordner in der Call Order gehören. Nachfolgend finden Sie die Attribute von Instanzen. Hier werden die Instanzattribute angezeigt, die zum ausgewählten Element im Abrufauftrag gehören. Wenn wir uns die Instanzattribute in unserem Beispiel genauer ansehen, stellen wir fest, dass diese Instanz zwei Attribute enthält. Das Kronenattribut enthält eine Referenz auf das Objekt. Wenn Sie darauf tippen, erscheint ein Fenster mit allen Attributen der primitiven Typen der Instanz. Attribut “kmen” zu einem bestimmten Zeitpunkt hat nirgendwo eine Verbindung. (zeigt Null an)

Wenn die lokalen Variablen, die auch zu dem aktuell ausgewählten Element im Abrufauftrag gehören, unten angezeigt werden, zeigt der Debugger hier nur Variablen an, die bereits über einen dedizierten Speicher und den zugewiesenen Startwert verfügen, d.h. sie sind definiert.

8. Ausnahmen

Bisher gingen wir davon aus, dass das gesamte Programm "ausfällt", sobald unvorhergesehene Ereignisse eintreten. Dieses Problem kann jedoch durch Ausnahmen vermieden werden. In solchen Situationen tritt eine Ausnahme auf, die den Programmablauf und den Übergang des Threads an die Stelle, an der die Situation behandelt wird, sofort unterbricht. Wenn das nicht der Fall ist, wird der Thread beendet und da wir nur einen Thread verwenden, wird die gesamte Anwendung abstürzen. Es gibt mehrere Arten von Ausnahmen, die nach der Ursache der unerwarteten Situation aufgeschlüsselt werden können:

- Fehler - kritischer Fehler, verursacht z.B. durch Ressourcenmangel für die virtuelle Maschine - OutOfMemoryError, Stapelüberlauf - StackOverflowError oder ähnliches. Diese Ausnahmen werden in der Regel nicht behandelt.

- RuntimeException - oft ein Fehler des Programmierers, (zB Null Division - ArithmeticException, ungültiger Index - ArrayIndexOutOf BoundsException). Um diese Bedingung aufzurufen, müssen wir nicht die Fähigkeit deklarieren, die Methode im Kopf aufzurufen.

- Ausnahme des Benutzers (z.B. anstelle einer Telefonnummer gibt er Buchstaben ein), oder eine nicht vorhandene Datei oder einer Datei eines anderen Typs. Auf solche Ausnahmen können wir oft sehr leicht reagieren. Lassen Sie beispielsweise den Benutzer die Datei noch einmal auswählen. Für solche Ausnahmen müssen wir immer das Schlüsselwort throws und die Liste der aufgerufenen Ausnahmeklassen angeben.

8.1. Geschützter Modus

Der potenziell problematische Teil des Codes kann mit Try-and-Compound-Klammern in den "protected mode" versetzt werden. Dieser Modus ist etwas langsamer. Tritt jedoch ein Fehler auf - eine Ausnahme -, wird der im Catch enthaltene Teil ausgeführt. Der optionale Block wird schließlich immer ausgeführt. Schließlich wird es oft für die Bereinigung von Job-Ausnahmen verwendet, wenn wir zu einer Ausnahme zurückkehren, wie z.B. das Schließen von Dateien, das Freigeben von Speicherplatz und so weiter.

```
public static void exception () {
      try {
            int a = 5 / 0; //create exception
      }
      catch (Exception e)
      {
            System.out.println("An exception was taken ");
      }
      finally
      {
            System.out.println("The contents of the block will
            always be processed.");
      }
}
```

8.2. Throw

Alternativ können wir die obige Ausnahme mit dem Schlüsselwort throw hinzufügen. Dies gilt nur für die geprüften Ausnahmen, falls sie die Ausnahmebehandlung an die aufrufende Methode übergeben wollen.
Wie dieses Beispiel zeigt:

```
if (index == null) {
      throw new NullPointerException();
}
```

8.3. Throws

Wenn wir eine Methode erstellen, die eine Ausnahme erzeugen kann, welche wir nicht behandeln wollen oder können. Wir teilen dem Übersetzer explizit mit, dass wir diese Ausnahme an die Top-Level-Behandlung mit der Ausnahme throws + class übergeben.

```
public static void  read ( ) throws IOException {
      ...
}
```

9. Arbeiten mit Dateien

Alle im Speicher gespeicherten Informationen, Daten oder Objekte werden beim Beenden des Programms gelöscht. Um sie zu bewahren und neu zu laden, müssen wir sie in einer Datei speichern.

Für eine reibungslose Dateneingabe ist es am besten, die Anwendungsdaten im Ordner appdata zu speichern. Die AppData oder Anwendungsdaten oder Anwendungsdaten enthalten Daten, die von Programmen erstellt wurden. In diesem Ordner erstellt es seinen eigenen Ordner praktisch jedes Programm, das auf dem Computer installiert ist, und speichert dann verschiedene Daten darin. Der einfachste Weg, um zu diesem Ordner zu gelangen ist, Dateien in den Roaming-Unterordner im Datei-Explorer %appdata% einzufügen, die Daten in diesem Ordner sollten den Benutzern auf verschiedenen Computern innerhalb der Domäne folgen.

Das Paket java.io enthält eine File-Klasse, die uns alle wichtigen Werkzeuge zur Dateiverwaltung zur Verfügung stellt. Viele Methoden der Klasse File benötigen als Argument einen Dateinamen. Als Argument können Sie die Zeichenkette String oder die Instanz der Klasse File verwenden. Dies ist oft die optimale Lösung, mit der wir einige Informationen finden oder im Voraus eine Operation über die Datei durchführen können.

Sie können ein Dateiobjekt auf verschiedene Arten anlegen:

- Der Name der Datei - wir erstellen ihn aus einem absoluten oder relativen Pfad, der in einen abstrakten Pfad umgewandelt wird.
- Der Name der Datei in Bezug auf den übergeordneten Ordner - der abstrakte Pfad wird in Bezug auf den übergeordneten Ordner erstellt.
- Uniform Resource Identifier (URI) - Bestimmte Anforderungen müssen erfüllt sein. Z.B. darf der Pfad nicht leer sein.

So kann beispielsweise die Datei selbst über die URI erstellt werden.

```
import java.io.File;
import java.io.IOException;
...
public void createFile() throws IOException{
      File soubor = new File("C:\\Temp\\hello.txt");
      soubor.createNewFile();
}
```

Das Dateisystem kann Einschränkungen für bestimmte Operationen am aktuellen Datei-

systemobjekt implementieren, wie z.B. Lesen, Schreiben und Booten, die wir Zugriffsrechte nennen.

Instanzen der Klasse File sind invariant, d.h. sobald die Instanz erstellt wurde, ändert sich der abstrakte Pfad, den das File-Objekt repräsentiert, verändert sich nie.

Die Dateiklasse bietet verschiedene Methoden zum Arbeiten mit Dateien. Zum Beispiel können wir arbeiten mit:

- Dateipfad - Das Dateiobjekt ist ein abstrakter Pfad zur Datei. Auf diese Weise können wir anders arbeiten. Rückgabemethoden, die den Pfad zu einer Datei zurückgeben, haben typischerweise einen Rückgabewert vom Typ Textzeichenkette, z.B. durch Verwendung von:
    - GetPath () ruft einen abstrakten Pfad zur Datei ab.
    - getName () Ermittelt den Namen der Datei.

- Dateiinformationen - Es gibt mehrere Methoden, die Dateiinformationen zurückgeben, wie z.B.: Länge (); canRead () oder, zum Beispiel, lastModified ().

- Verzeichnisinformationen - Wir haben mehrere Methoden nur für das Verzeichnis, wie z.B. list () - eine Liste aller Dateien im Verzeichnis oder Unix SheetRoots () Rückgabefelder aller Verzeichnisbaum-Root.

- Arbeiten mit Dateien - Wir haben verschiedene Methoden, um mit Dateien zu arbeiten:
    - RenameTo (File k) als Parameter gibt die nächste Instanz des File-Objekts an.
    - löschen ()
    - mkdir () Verzeichniserstellung
    - setReadOnly ()

10. Grafische Benutzeroberfläche (GUI)

Es ist die grafische Umgebung, mit der der normale Benutzer konfrontiert wird und mit der er arbeitet. Wir alle kennen die einzelnen Komponenten dieser Umgebung sehr gut. Dazu gehören beispielsweise Knöpfe, Scroller, Schieberegler und dergleichen. Verschiedene Grafikbibliotheken wie AWT (Abstract Windowing Toolkit), JFC (Java Foundation Classes) sind in Java verfügbar. Für BlueJ können Sie die Simple GUI Designer Extension installieren. Das kann die GUI-Erstellung vereinfachen. In diesem Beitrag wird diskutiert, wie man mit JFC, auch bekannt als Swing, arbeitet.

10.1. Erstanwendung

Beim Erstellen einer Anwendung mit einer grafischen Benutzeroberfläche, in der die Anwendung ausgeführt wird müssen wir zuerst ein "Fenster" (Rahmen) schaffen. Wir werden die JFrame-Klasse verwenden. Es gibt mehrere Möglichkeiten, ein Fenster zu erstellen. Wir haben die ConstructGui-Klasse erstellt, die von der JFrame-Klasse erbt. In dieser Klasse haben wir einen nicht-parametrischen Konstruktor erstellt. Wir haben einen Knopf und ein Label in der Klasse platziert.

Eine weitere gängige Komponente, die wir in diesem Beispiel nicht verwendet haben, ist das Schreibfeld (JTextField). Es würde wie folgt deklariert werden:

```
JTextField someName = new JTextField(”Text”, 6)
```

Geben Sie als Parameter den Text ein, der im Feld angezeigt wird. Als nächstes gibt es eine Zahl, die angibt, wie viele Zeichen das de Feld passen soll.

Wir haben das Layout der Komponenten in FlowLayout festgelegt. Aufgrund von FlowLayout war es notwendig, java.awt zu importieren.

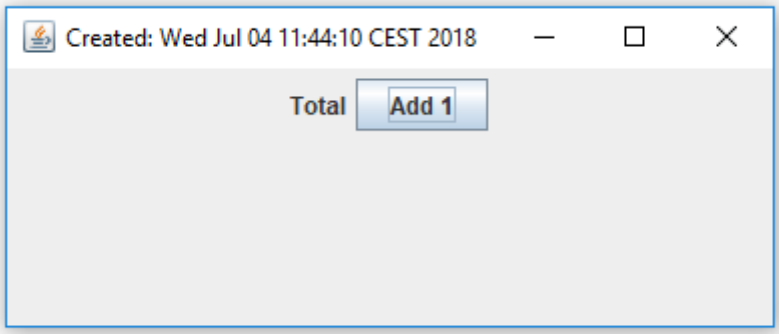
Die einzelnen Komponenten mussten dem Fenster hinzugefügt werden. Was wir mit der add () Methode gemacht haben, die den Namen des hinzugefügten Objekts als Parameter angibt.

```
import java.awt.*;
import javax.swing.*;

public class ConstructionGui extends JFrame {
      private JButton button;
      private JLabel value;

      public ConstructionGui()
      {
            FlowLayout layout = new FlowLayout();
            setLayout(layout);
            value = new JLabel(”Total:”);
            add(value);
            button = new JButton(”Add 1”);
            add(button);
      }
}
```

Anschließend haben wir eine weitere Klasse mit dem Namen FirstGUI erstellt. Mit der Methode main. In der Methode main haben wir Objekte mit der Klasse ConstructionGUI erstellt. In diesem Objekt haben wir verschiedene grundlegende Methoden ausgewählt. Wir haben eine Bibliothek von Daten importiert, um die Funktion date() zu verwenden, die uns das heutige Datum in das Titelfenster schreibt.

```
import java.awt.*;
import javax.swing.*;

public class ConstructionGui extends JFrame {
      private JButton button;
      private JLabel value;

      public ConstructionGui()
      {
            FlowLayout layout = new FlowLayout();
            setLayout(layout);
            value = new JLabel(”Total”);
            add(value);
            button = new JButton(”Add 1”);
      add(button);
}}
```

Das Ergebnis sieht so aus:



Die einzelnen Komponenten sind zentriert. Wenn wir das Fenster skalieren, werden sie untergeordnet.

11. Ereignisse

Jetzt haben wir ein einfaches Single-Label-Fenster erstellt. Es passiert jedoch nichts, wenn die Taste gedrückt wird. Deshalb verwenden wir Ereignisse, um unserem Fenster Funktionalität hinzuzufügen. Das Prinzip der Ereignisse ist, dass wir in der GUI z.B. ein Ereignis auf dem Taster aufrufen. Das Ereignis, für das wir das Ereignis erstellt haben, ist ein Ereignis-Listener. Die Methode, etwas zu tun, wird beim Ereignis-Listener aufgerufen. In unserem Fall erstellen wir die Schaltflächen Button und Button Counter aus dem Label. Der vollständige Code befindet sich auf der nächsten Seite.

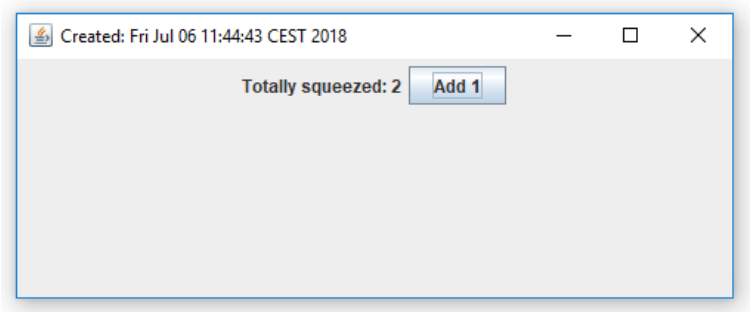
Wir haben der ConstructionGui-Klasse eine weitere Klasse namens EventCost hinzugefügt. Dadurch hat es Zugriff auf ConstructionGui-Komponenten. Wir haben den ActionListener in der Klasse EventCost implementiert. Es ist notwendig, java.awt.event zu importieren. *;

Die ActionListener-Klasse enthält nur eine Rückgabemethode actionPerformed (ActionEvent e), in der wir definieren, was passiert, wenn ein Ereignis aufgerufen wird. In unserem Fall ist das Zählen ein Knopfdruck.

Im Konstruktor haben wir Ereignishörer erstellt - ein Objekt namens MyCount. Wir haben der Nummer des Zuhörers einen Button zugeordnet.

```
EventCost MyCount = new EventCost();
MyButton.addActionListener(MyCount);
```

Das Ergebnis sieht so aus:



```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class ConstructionGui extends JFrame {
      private JButton MyButton;
      private JLabel MyValue;
      int MyNumber = 0;

      public ConstructionGui()
      {
            FlowLayout layout = new FlowLayout();
            setLayout(layout);
            MyValue = new JLabel("Not squeezed ");
            add(MyValue);
            MyButton = new JButton("Add 1");
            add(MyButton);
            EventCost MyCount = new EventCost();
            MyButton.addActionListener(MyCount);
      }

      public class EventCost implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                  MyNumber = MyNumber + 1;
                  MyValue.setText("Totally    squeezed:    "    +
                  MyNumber);
            }
      }
}
```

EINFÜHRUNG IN DIE MEDIENWISSENSCHAFTEN

1. Einführung in die Medienwissenschaften

Anmerkung: Das zentrale Thema dieses Lehrfachs ist die Medienkommunikation als untrennbarer Teil der modernen und postmodernen Gesellschaft. Die Einführung in die Medienlehre beschäftigt sich mit den grundlegenden Etappen der Entwicklung der menschlichen Kommunikation, wobei die Rolle der Massenkommunikation und ihrer Auswirkungen auf die Öffentlichkeit akzentuiert wird – die EmpfängerInnen medialer Informationen. Der interdisziplinäre Sachverhalt wird hinsichtlich der Beziehungen zu ihrer historischen Dimension analysiert.

2. Grundkonzepte: Medium, Mediation, Medialisierung

Medialisierung: Sozialer Wandel, die Grundlage für die noch nie dagewesene Verbreitung von Kommunikationsmedien und deren zunehmend an Bedeutung gewinnenden Rolle im sozialen Leben.

Merkmale der Medialisierung:

- **Erweiterung:** Medien erweitern die Möglichkeiten der menschlichen Kommunikation
- **Ersatz:** Medien ersetzen ein paar soziale Aktivitäten (TV-Debatten ersetzen Treffen vor Wahlen)
- **Verschmelzung:** allmählich verschwinden die Grenzen zwischen medialen und nichtmedialen Aktivitäten – die mediale Definition von Realität vermischt sich mit der sozialen Definition von Realität zu einem großen Ganzen.
- **Einrichtung:** Medien sind ein wichtiger Industriezweig

Mediation: Der Prozess, während welchem ein Vermittler zwischen zwei Parteien auftritt, um deren Beziehung zueinander zu beeinflussen oder sicherzustellen.

Medium: Mittel; Umwelt; es vermittelt einen Vorgang

Medium: (Im Bereich der Medienwissenschaften) stellt ein Medium eine wichtige Ver-

knüpfung zwischen dem/der SenderIn und dem/der EmpfängerIn dar, zB zwischen den RedakteurInnen einer Zeitung, eines Journals, eines Radio- oder Fernsehsenders oder einer Online-Redaktion und deren LeserInnen, ZuhörerInnen, ZuschauerInnen oder TeilnehmerInnen an Online-Diskussionen usw.

Medien sind Mittel der Massen- oder Medienkommunikation, welche Informationen in verschiedener Form und zu unterschiedlichen Zwecken übermitteln. Sie lassen sich grob einteilen in:

- Printmedien (Zeitungen, Journale etc.)
- Elektronische Medien (Radio, TV)
- Neue Medien (Internet, Soziale Netzwerke wie Facebook, Instagram usw.)

3. Kommunikation, Gesellschaft, Medien, Phasen in der Entwicklung der menschlichen Kommunikation

Hauptphasen in der Entwicklung der menschlichen Kommunikation:

- Ära der Zeichen und Laute
- Ära des Sprechens und der Sprache
- Ära des Schreibens
- Ära des Buchdrucks
- Ära der Massenkommunikation
- Ära der Computer und Netzwerkmedien

Marshall McLuhan (1911–1980) gliedert die menschliche Kommunikation in folgende Epochen:

- *Epoche der mündlichen Stammeskultur:* Zeit des akustischen Raums
- *Epoche der geschriebenen Kultur:* die akustische Wahrnehmung wird durch die visuelle Wahrnehmung ersetzt
- *Gutenberg-Galaxie:* ein gedrucktes Buch ist das erste Massenerzeugnis, die erste reproduzierbare Art von Konsumgut
- *Marconi-Galaxie:* jenes Zeitalter, welches mit dem Siegeszug der Elektrizität in Verbindung gebracht wird

Die heutige Epoche der digitalen Computernetzwerke lässt sich als Galaxie der Torwege bezeichnen.
Werner Faulstich wandelt auf den Spuren von McLuhan und legt die Phasen der

menschlichen Kommunikation wie folgt fest:

- *Phase A:* Beginn der Vorherrschaft der Medien: Menschen übernehmen bis um das Jahr 1500 herum die Rolle von Medien.

- *Phase B:* zweite Welle, Vorherrschaft der (Print-)Medien: In der Zeit zwischen 1500 und 1900 entwickeln sich Printmedien vom anfänglichen exklusiven Gut hin zu einem Massenprodukt.

- *Phase C:* Verlagerung des Fokus auf tertiäre (elektronische) Medien: Zwischen 1900 und 2000 kommen neben Zeitungen auch das Radio und später das Fernsehen auf

- *Phase D:* Verlagerung des Fokus auf digitale Medien: Die Zeit von 2000 bis heute, Trend geht weg von der Breitenstreuung hin zur gezielten, individuellen Ansprache

Im Jahr 1995 veröffentlichte Mark Poster das Buch „The Second Media Age", wo er die Entwicklung der Medien in zwei Phase unterteilt:

- *Erstes Medienzeitalter:* Bezeichnend hierfür sind Übertragungstechnologien (Senden), von einem zentralen Punkt hin zu Peripherien mit einer hohen Übernahme- und einem geringen Wechselseitigkeitsgrad ("Senden" bedeutet in diesem Zusammenhang die Verteilung von Informationen ausgehend von einer Quelle hin an viele EmpfängerInnen, sprich auch die Verteilung von Zeitungen)

- *Zweites Medienzeitalter:* Dieses wird durch die Etablierung von Kommunikationsnetzwerken begründet. Das Prinzip der Breitenstreuung wird durch das Prinzip der Interaktion zwischen individuellen Knotenpunkten von Netzwerken mit einem geringen Maß an Übernahme und einem hohen Maß an Wechselseitigkeit ersetzt.

4. Typologie sozialer Kommunikation

Grundlegende Parameter zur Unterscheidung der Arten von sozialer Kommunikation:

- Ausmaß der Individualisierung oder Sozialisierung der Kommunikationsaktivität
- Ausmaß der Institutionalisierung
- Anzahl der TeilnehmerInnen
- Verhältnis der Beziehung von Gleichheit zu Ungleichheit

Arten sozialer Kommunikation

- *Intrapersonelle Kommunikation* (Selbstgespräche)

- *Interpersonelle Kommunikation* (zwischen zwei oder drei Individuen): Die Beteiligten befinden sich sowohl im selben Situations- als auch im selben Kommunikationskontext, die Rolle der SprecherInnen und ZuhörerInnen verschwindet, die AkteurInnen werden als Individualisten wahrgenommen

- *Gruppenkommunikation* (innerhalb einer Gruppe mit einer bestimmten internen Hierarchie: Familie, Freunde, eine kleine Arbeitsgruppe. Eine höhere Autorität beteiligt sich an der Kommunikation, steuert diese und die anderen respektieren das).

- *Kommunikation unter Gruppen* (zwischen/unter gebildeten Gruppen: Schulklassen, Sportvereine, höherer Formalisierungsgrad)

- *Organisationskommunikation* (innerhalb der Gesamtheit einer Organisation, zB einer Schule oder einer politische Partei): Die Möglichkeiten der Dialogführung sind begrenzt, die Beziehung zwischen den einzelnen AkteurInnen ist nicht gleich.

- *Gesellschaftliche Kommunikation* (alle verfügbaren Kommunikationsprozesse der Mitglieder einer bestimmten Gesellschaft): Der Dialog-Charakter geht verloren, die Kommunikation erfolgt im Wesentlichen in eine Richtung, der Individualisierungsgrad ist geringer, die Anonymität größer.

Gesellschaftliche Kommunikation

Hiervon werden zwei Arten unterschieden:

- Öffentliche Kommunikation (Vorlesung, politische Veranstaltung, Einheit von Zeit und Ort)

- Medienkommunikation (eine historisch bedingte Art der Medienkommunikation ist die Massenkommunikation)

5. Massenmedien, ihre Kennzeichen, Funktion und Entwicklung

Charakteristische Merkmale von Massenmedien

- Nachrichten sind für die kurzzeitige Verwendung bestimmt, sie haben aktuellen Charakter
- Anonymes Massenpublikum
- Uneingeschränkte, öffentlich zugängliche Information für jeden
- Informationsfluss hauptsächlich in eine Richtung
- Verzögerte Rückmeldung
- Informationsperiodizität
- Informationen werden ständig und regelmäßig angeboten

Funktionen von Massenmedien

- Informieren, Bereitstellung von Informationen über Ereignisse
- Sozialisieren, Erklären und Kommentieren von Ereignissen, Unterstützen von Behörden und Aufrechterhalten sozialer Standards, Festlegen einer Prioritätenreihenfolge
- Kontinuität, Unterstützung vorherrschender kultureller Schemas
- Unterhalten, Anbieten von Aufregung und Vergnügen
- Gewinnung und Mobilisierung von Menschen für wichtige soziale Anliegen

Entwicklung von Massenmedien

Das Aufkommen der Massenmedien ist eng verbunden mit:

- der Entwicklung der technischen Erzeugungsmöglichkeiten für große Mengen an Druckausgaben mit identischem Inhalt in einem relativ kurzen, bekannten und regelmäßigen Zeitabstand (Veröffentlichung periodischer Druckwerke, Vorführen von Filmen, Rundfunk-programme).
- sozialen Bedingungen für deren Verwendung (EmpfängerInnen verändern sich)
- ihrer wirtschaftlichen Attraktivität

Epochen der Entwicklung von Massenmedien

- *Beginn des 19. Jahrhunderts:* Entwicklung der Massenpresse
- *Beginn des 20. Jahrhunderts:* Aufkommen des Bewegtbilds
- *20er- und 30er-Jahre des 20. Jahrhunderts:* Beginn der flächendeckenden Radioübertragung (in der Tschechischen Republik: 1923)
- *50er und 60er-Jahre des 20. Jahrhunderts:* Beginn der flächendeckenden TV-

Übertragung (in der Tschechischen Republik: 1953)
- *1990er-Jahre bis heute:* Aufkommen und Entwicklung digitaler Medien

6. EmpfängerInnen

EmpfängerInnen: institutionalisiertes NutzerInnenkollektiv oder AdressatInnen einer Nachricht

Phasen der EmpfängerInnen-Entwicklung:

- Elitäre Leserschaft: Entstehung einer Leserschaft, geringe Anzahl gebildeter LeserInnen

- Massenleserschaft: bildet sich in der ersten Hälfte des 19. Jahrhunderts aus der Leserschaft der Gossenpresse heraus und existiert bis heute, wobei auch TV-ZuschauerInnen dieser Kategorie zugerechnet werden.

- Special Interest-Leserschaft: Entstand infolge der Entwicklung von Kunst und Wissenschaft (Fachjournale, wissenschaftliche Journale für Interessensgruppen, Radio- und TV-Stationen)

- Interaktive Leserschaft: bildete sich infolge des Aufkommens neuer Medien (Internet und soziale Netzwerke).

Die ersten MedienkonsumentInnen waren LeserInnen. Mit der Erfindung des Buchdrucks Mitte des 15. Jahrhunderts ging die Möglichkeit zur Aufzeichnung und Vervielfältigung ein- und derselben Nachricht einher. Mit einem Mal waren die LeserInnen mit einer großen Anzahl an Kopien derselben Nachricht konfrontiert. Die Leserschicht war jedoch nicht besonders zahlreich, die Menge an Drucker-pressen war begrenzt und Bücher waren teuer und schwer zu bekommen. Unter diesen Bedingungen war die Herausbildung einer Massenleserschaft nicht möglich. Es fehlte schlichtweg die Möglichkeit, das mediale Angebot weit und in regelmäßigen Zeitabständen zu verbreiten.

Durch die zunehmende Alphabetisierung der Bevölkerung entstand auch eine neue Art von Öffentlichkeit: Ein Teil des Adels, der Bürgerschaft und der aufkommenden Bourgeoisie begann damit, sich für die Durchsetzung ihrer politischen Anliegen zu interessieren. Der politische Diskurs wanderte damit vom privaten in den öffentlichen Raum. Die Entstehung der Öffentlichkeit wurde zu einer Voraussetzung für das Aufkommen von Massenmedien und deren KonsumentInnen, welche Medien (gedruckte Materialen) benötigten, um Antworten auf ihre Fragen zu finden. Medien brauchen die Öffentlichkeit, damit die verbreiteten Informationen auch EmpfängerInnen finden. Gleichzeitig braucht die Öffentlichkeit Medien, um dort relevante Themen platzieren und diskutieren zu kön-

nen.

Konzepte von RezipientInnen

- Konzept passiver RezipientInnen
- Konzept aktiver RezipientInnen
- Konzept interaktiver RezipientInnen

7. Auswirkungen von Medien und ihrer Phasen

- Bedenkt man die vermeintlichen Auswirkungen von Medien, ist es notwendig, sich ihres grundlegenden Einflusses gewahr zu werden: Der Einfluss von Medien auf das Individuum ist klar, aber diesen zu erklären ist ebenso schwierig, wie ihn eindeutig nachzuweisen.

- Darüber hinaus ist es notwendig, sich darüber im Klaren zu sein, dass ein und derselbe Medieninhalt vollkommen unterschiedliche Auswirkungen haben kann, sprich bei verschiedenen Individuen unterschiedliche Wahrnehmungen und Reaktionen auslösen wird.

Kriterien für die Klassifizierung vermeintlicher Medienauswirkungen (nach Watson, 1998):

- Was wird beeinflusst?
- In wem?
- In welchem Ausmaß?
- In welchem Zeitabschnitt?

Basierend auf diesen Fragen, können die folgenden Medieneffekte unterschieden werden:

- *Kurzfristige und langfristige Effekte:*
  - unmittelbare Reaktionen auf mediale Reize, zB einen Bericht über Terrorismus, den Sturz einer Regierung, einen Olympiasieg.
  - Dem gegenüber stehen langfristige Veränderungen individueller Sichtweisen, zB betreffend die Gesellschaftsordnung, welche auch durch Medien ausgelöst werden können.

- *Direkte und indirekte Effekte:*
  - Direkte Effekte sind sehr schwer nachzuweisen, bemerkbar machen sie jedoch, wenn man zB die Wahlkampagne eines Präsidentschaftskandidaten mit der Abschneiden bei der Wahl vergleicht. Ein weiterer direkter Effekt ist der Einfluss einer Werbe-kampagne auf das Kaufverhalten von KonsumentInnen, welcher sich durch eine messbare Absatzsteigerung infolge dieser Kampagne belegen lässt.
  - Indirekte Effekte manifestieren sich erst mit beträchtlicher Zeitverzögerung und zwar im Zusammenspiel mit anderen Faktoren.

- Geplante und ungeplante Effekte:

- Zu geplanten Effekten zählen insbesondere: kommerzielles, politisches und soziales Marketing und Öffentlichkeitsarbeit
- Unter ungeplante Effekte fällt zB aggressives Verhalten von TV-ZuschauerInnen infolge der Ausstrahlung gewalttätiger Filme usw.

Funktionen der Medienwirkung:

- *Kognitiver Natur (Verstehen):* Medien schaffen Anreize zum Lernen, zB geografisch-kognitive Programme, Programme über die Welt der Wissenschaft und Technologie, Sprachkurse usw.

- *Gefühlsbezogen:* insbesondere Filme und Serien wecken Emotionen wie Freundlichkeit und Zärtlichkeit, erzeugen jedoch auch Angst und Stress.

- *Physiologischer Natur:* das Anhören von Musik oder das Ansehen eines Films oder einer Serie, kann dazu führen, dass man sich entspannt oder aufregt

- *Verhaltensbezogen:* hierbei handelt es sich vor allem um Veränderungen betreffend das Konsumverhalten von Menschen

- *Wertebezogen (konstruktiv oder destruktiv):* Respekt vor schwächeren, beeinträchtigten Personen und der Wunsch, ihnen zu helfen. Auf der anderen Seite kann auch der Wunsch stehen, Konflikte gewaltsam zu lösen.

Im Jahr 1999 klassifizierte McQuail vier Phasen der Medienwirkungsforschung:

- *Uneingeschränkte Macht der Medien (1900 – 1940):* unmittelbare Auswirkungen von Medien-inhalten, Medieninhalte führen zu einem identischen Effekt, sprich wirken sich exakt gleich auf die EmpfängerInnen aus.

- *Ineffektivität von Medien (1940 – 1965):* Die individuellen Persönlichkeitsmerkmale verändern sich, wodurch auch der Medienkonsum selektiver wird.

- *Neuer Glaube an starke Medienwirkung (1965 – 1980):* aktive Haltung der RezipientInnen zu Medien

- *Transaktionsvorstellung von Medieneffekten (1980 bis heute):* starke Position von Medien, aber gleichzeitig auch eine starke Position der Öffentlichkeit

8. Medien und Macht, Propaganda

Macht (soziologische Definition): Ausdruck für eine höhere Position in der sozialen Beziehung (jemanden dazu bringen, etwas anderes zu tun, als er oder sie wollte). Wenn es keine Möglichkeit der Einschränkung gibt, spricht man oft von Einfluss.

John B. Thompson hat **vier Arten** der Ausübung von Macht in der Gesellschaft in seinem Buch "Media and Fashionability. Social Theory of Media" (1995) festgehalten:

- *Wirtschaftliche Macht:* ergibt sich aus den Mitteln, Wohlstand zu schaffen
- *Politische Macht:* Entscheidungsautorität bedingt durch ein Wahlergebnis oder eine anderweitig legitimierte Besetzung einer politischen Funktion
- *Bestrafungsmacht:* ergibt sich durch die Ausübung der Macht der Bestrafung
- *Symbolische Macht:* ergibt sich aus der Möglichkeit, mittels Worten, Bildern oder Tönen Unterstützung für die Machtausübung zu generieren oder zu mobilisieren

Beziehung zwischen Macht und Medien (zwei Haltungen):

- Medien haben eine derart starke Position in der Gesellschaft, dass ihre Funktion als Machtausübung angesehen wird; die Medienfunktion ist innerhalb bestehender staatlicher Strukturen angesiedelt, aber nicht wirtschaftlich damit verbunden. Wenn die regierende Klasse in der Lage ist, Werkzeuge zur Kontrolle von Medien zu schaffen und durchzusetzen, erhält sie dadurch eine derartige Dominanz über die Medien und die Öffentlichkeit, dass dadurch die bereits vorhandene Macht zusätzlich gestärkt und gefestigt wird.
- In der Beziehung zwischen Medien und Macht ist es möglich, auch positive, ja fast sogar heilbringende Merkmale zu entdecken. Medien dienen als Kontrollorgan für die Demokratie, das bedeutet, dass sie die Kontrollmacht über die mit der Regierung und der Gesetzgebung betrauten Personen haben (in der Tschechischen Republik sind dies der Präsident, die Regierung und das Parlament).

Propaganda: überzeugendes Format (überzeugende Kommunikation). Es handelt sich hierbei um die beabsichtigte Manipulation des Denkens und Verhaltens mithilfe von Symbolen. Propaganda ist im Wesentlichen strategisch geplante Kommunikation. Sie hat einen offensiven Charakter, ist langfristig ausgerichtet und konzeptionell, zielt darauf ab, die Weltanschauung zu verändern, das Bewusstsein einer Gruppe oder einer ganzen Gesellschaft an einem Wunschzustand auszurichten und bestimmte Verhaltensmodelle zu etablieren.

Arten von Propaganda:

- *Politische:* konzentriert sich darauf, Macht zu bekommen und zu erhalten
- *Wirtschaftliche:* forciert den Kauf und den Verkauf von Gütern sowie die Aufrechterhaltung des Vertrauens in das Wirtschaftssystem.
- *Kriegerische (Militär):* Demoralisieren des Feinds oder moralische Unterstützung der eigenen Armee und Bevölkerung
- *Diplomatische:* stärken von Freundschaften (Feindschaften) und Verbündeten (Gegnern)
- *Ideologische:* Verbreiten eines umfassenden Systems von Ideen
- *Didaktische:* Art der Bevölkerungserziehung, Durchsetzen sozial wünschenswerter Ziele
- *Eskapistische:* spezielle Form der politischen Propaganda, welche Medien einsetzt, um die Aufmerksamkeit von sozialen Problemen abzulenken.

9. Typologie von Printmedien

Printmedien: Alle Medien, deren Inhalt an Papier gebunden ist, zB Flugblätter, Zeitungen, Journale, Bücher usw.

Typologie-Kriterien:

- basierend auf der Abdeckung der Leserschaft
- basierend auf Periodizität
- basierend auf Inhalten

Abdeckungsbasiert: Printmedien werden in einem bestimmten, festgelegten Gebiet verteilt. Man unterscheidet zwischen:

- Lokalen Medien (Gemeindeblatt, ländliche Zeitungen)
- Regionalen Medien (Daily from České Budějovice Region, Voice of Vysočina)
- Überregionalen Medien (Mitteilungsblatt von Šumava)
- Bundesweiten Medien (Lidové noviny, Hospodářské noviny/Wirtschaftszeitung)
- Internationalen Medien (Reader´s Digest)

Periodizitätsbasiert: Periodizität wird im Gesetz für Rechte und Pflichten für die Veröffentlichung periodischer Druckwerke definiert.

Periodische Presse: Alle Zeitungen, Magazine und andere gedruckten Materialien, welche unter demselben Namen, mit derselben inhaltlichen Ausrichtung und einem einheitlichen grafischen Layout zumindest zweimal pro Kalenderjahr verlegt werden.

Zeitungen: Gedruckte Medien (regelmäßig), welche zumindest zweimal pro Woche ver-

legt werden und einen aktuellen politischen Teil enthalten, welcher durch mannigfaltige Themen (Diversität) gekennzeichnet ist.

Magazin: Gedrucktes Medium, welches in längeren Intervallen als Zeitungen aufgelegt wird, maximal einmal pro Woche und mindestens zweimal pro Jahr.

Gesetzessammlungen und amtliche Bekanntmachungen dürfen nicht als regelmäßige Printmedien betrachtet werden.

Inhaltsbasiert:

- Wöchtenliche Newsletter (Týden/Week, Instinkt/Instinkt)
- Soziales und Lifestyle (Květy, Vlasta, Xantypa)
- Kinder-/Jugendmedien (ABC, Bravo, Dívka/Mädchen)
- Interessen und Hobbys (Golf, Tennis, Radfahren, Bienenzucht)

10. Vier Pressetheorien

Im Jahr 1956 formulierten die Medienanalytiker Friedrich Siebert, Theodor Peterson und Wilbur Schramm im Artikel "Vier Theorien der Presse" vier Pressetheorien.

Es gibt vier Einstellungsarten, nach welchen sich die Beziehung zwischen der Gesellschaft (politischen Herrschaft) und Medien auflösen lässt:

- **Autoritätstheorie:** Medien dienen als Mittel, um die Standpunkte und Meinungen einer Autorität (z.B. HerrscherIn oder ein/e die Medien kontrollierende/r PolitikerIn) zu vermitteln, welche mit der aktuellen Machtteilung übereinstimmen.

- **Liberalitätstheorie:** Gleichbedeutend mit der Theorie der freien Presse. Ihr zufolge verhindern die Medien, dass der Staat die öffentliche Kommunikation kontrolliert – alle medial vermittelten Meinungen sind ausgeglichen. Medien sind in einer Art und Weise organisiert, welche es ihnen ermöglicht, jederzeit zu sagen, was sie wollen.

- **Theorie der sozialen Verantwortung:** Medien sollten bestehende Probleme aus mehreren Perspektiven zeigen, allerdings sollte es auch Einschränkungen geben, sprich Grenzen, die nicht überschritten werden dürfen (z.B. keine Unterstützung von Gewalt, Kriminalität, Terror usw.)

- **Sowjetisch-kommunistische Medientheorie:** Medien sind das Instrument einer Art von Sozialisierung und öffentlicher Meinungsbildung, sie dienen als Mittel zur Bildung und zur Erziehung der Öffentlichkeit (Erziehung sozialistischer BürgerInnen)

Denis McQuail schlug ergänzend zwei weitere Konzepte vor:

- **Entwicklungsfördernde Medientheorie:** Medien sollten zur Modernisierung der Gesellschaft beitragen und dazu dienen, Sozialisierungs- und Erziehungsziele zu erreichen.

- **Theorie der demokratischen Partizipation:** Medien repräsentieren eine soziale Institution, es soll keine demokratische Medienkontrolle geben, Medien sollten so ausgerichtet sein, dass sie die Interessen von Minderheiten und Individuen wahren.

11. Öffentlich-rechtlicher Rundfunkt

Folgende TV- und Radioorganisationen gehören zu Medien mit Rundfunk-Signal:

- Stationen des öffentlichen Sektors (staatlicher oder öffentlich-rechtlicher Rundfunk)
- Stationen des privaten Sektors

Im **öffentlichen Sektor** unterscheidet man den Bereich der Rundfunkmedien nach dem Grad der Verbundenheit mit dem Staatsapparat und mit der Regierung:

- Staatliches Radio und Fernsehen sind direkt der Weisungsbefugnis des Staatsapparates unterstellt: In Ländern mit autoritären Regimen dienen diese einzig und allein und uneingeschränkt dem jeweiligen Regime.

- Öffentlich-rechtliches Radio und Fernsehen werden gesetzlich geregelt: Es handelt sich um nicht-staatliche Organisationen, welche einen nicht gewinnorientierten Dienst für die Öffentlichkeit erbringen.

Der **private Sektor** im Bereich Rundfunkmedien besteht aus Sendeanstalten von Privatunternehmen: Die ersten Vertreter davon tauchten in den USA auf und existieren bis heute vor allem dort.
In Europa begann die Radioübertragung als lizensiertes Privatgeschäft: Als sich das Radio massiv verbreitete, verstaatlichten die einzelnen Ländern in Europa dessen Übertragung. So übernahm z.B. die Tschechoslowakei im Jahr 1925 mehrheitlich das Unternehmen Radiojournal in Tschechien. Auch die private BBC (British Broadcasting Company) wurde 1927 zu einer öffentlichen BBC-Gesellschaft (British Broadcasting Corporation). Damit kam ihr die Rolle als Pionier zu: Anders als andere europäische Radios wurde es nämlich kein Teil des staatlichen Apparats, sondern blieb eine öffentliche, nicht an staatliche Weisungen gebundene Medieninstitution.

Modelle des öffentlich-rechtlichen Rundfunks entwickelten sich in den westeuropäischen Demokratien nach dem Zweiten Weltkrieg, in Osteuropa erst nach 1989. Das öffentlich-rechtliche Radio und Fernsehen behielten in der Tschechischen Republik ihr Monopol: Einschränkungen des Spektrums der Oszillationsfrequenz verhinderten das Aufkommen privater Sender, in diesem Kontext notwendige Investitionen und Anforderungen an den Betrieb von Rundfunkeinrichtungen hatten jedoch ebenfalls einen Anteil daran.

In Westeuropa begann das Aufstreben privater Radiosender in den 70er-Jahren des 20. Jahrhunderts, die Entwicklung des Privatfernsehens nahm in den 1980er-Jahren an Fahrt auf. In dieser Zeit bildete sich ein **duales Rundfunksystem** heraus: öffentlich-rechtliches und privates Radio und Fernsehen existieren nebeneinander.

12. Medienerziehung und Medienbildung

Medienbildung: Wissen und Fähigkeiten, welche es Individuen ermöglichen, diverse Medienaspekte (Medieninhalte) zu verstehen und sich kritisch damit auseinanderzusetzen.
Medienerziehung: systematisches Vermitteln von Medienwissen.

Stufen der Medienbildung:

- **Medienerziehung:** Schule (siehe die fachübergreifenden Themen der Programme zur Allgemeinbildung in der Grundschule und Unterstufe); außerschulisch (Interessenzirkel junger JournalistInnen).

- **Professionelle Erziehung:** Ausbildung von Lehrkräften und zukünftigen Lehrkräften, welche aktuell oder zukünftig Medienerziehung unterrichten werden; Ausbildung von JournalistInnen.

- **Öffentliche Erziehung betreffend Medien, Medienkritik:** Übertragungen betreffend Medien in Radio und Fernsehen: Beurteilung der Angemessenheit von Fernseh-Programmen (z.B. ein Programm ist nicht für Kinder geeignet).

13. Bedeutende Persönlichkeiten der tschechischen Massenmedien

Josef Kajetán Tyl (1808 – 1856)

- Schriftsteller, Bühnendichter, Journalist
- 1833 gründete er das Magazin Jindy a nyní (Einst und jetzt), welches er später in Květy české (Böhmische Blüte) umbenannte. Er wollte ein tschechisches Magazin herausbringen, welches keine Kopie ausländischer Ausgaben war.
- Sein Magazin Květy wird bis heute verlegt.
- Er setzte sich für die öffentliche Erziehung und vor allem für die öffentlichkeitswirksame Aktivierung der ländlichen Leserschicht ein.

Karel Havlíček Borovský (1821 – 1856)

- 1846: Redakteur der Pražské noviny (Prager Zeitung)
- 1848: Gründung der ersten tschechischen Tageszeitung Národní noviny (Nationalzeitung)
- 1849: Gründung des Magazins Slovan (der Slawe), da Národní noviny aus politischen Gründen eingestellt wurde. Darin kritisierte er auch öffentlich die politische Situation im Kaisertum Österreich.
- 1851: Exil in Brixen (Schweiz)
- 1855: Rückkehr nach Böhmen
- Vertreter des Austroslawismus: Kooperation slawischer Nationen mit der Habsburger Monarchie
- Kompromissloses, kritisches Denken, logisches Argumentieren, kultivierte die tschechische Sprache

Julius Grégr (1831 – 1896)

- Tschechischer Politiker und Journalist
- 1862: Inhaber und Redakteur der Zeitung Národní listy; Es gelang ihm, darin die zukünftige Elite des tschechischen Journalismus zusammenzubringen. So publizierten zB Jakub Arbes und Jan Neruda in der Národní listy.

František Gel (1901 - 1972)

- 1924: Redakteur der Lidové noviny (Volkszeitung)
- 1945: Redakteur der politischen Sendung des tschechoslowakischen Radios, Reporter beim Nürnberger Prozess
- 1955: Vortragender für Journalismus an der Fakultät der Künste der Karlsuniver-

sität

Pavel Tigrid (1917 – 2003)

- Schriftsteller, Journalist, Politiker
- Vertreter des tschechischen Anti-Kommunisten-Exils
- 1939: Emigration nach England, Mitarbeit im Exil an BBC-Sendungen in London
- 1945: Rückkehr nach Prag, Chefredakteur des Magazins Obzory (Horizonte)
- 1948: erneute Emigration, Redakteur beim Radiosender Freies Europa, Redakteur beim Magazin Svědectví (Beweis)
- Nach 1989 kehrte er in die Tschechoslowakei zurück und wurde unter Präsident Václav Havel Kulturminister

14. Literatur

JIRÁK, J., KÖPPLOVÁ, B. 2009. Masová média. Praha: Portál, ISBN 978-80-7367-466-3

JIRÁK, J., KÖPPLOVÁ, B. 2003. Média a společnost. Praha: Portál, ISBN 80-7178-697-7

KONČELÍK, J. a kol. 2010. Dějiny českých médií v datech. Praha: Portál, ISBN 978-80-7376-698-8

OSVALDOVÁ, B., HALADA, J. 2007. Praktická encyklopedie žurnalistiky a marketingové komunikace. Praha: Libri, ISBN 978-80-7277-266-7

REIFOVÁ, I. a kol. Slovník mediální komunikace. Praha: Portál, 2004, ISBN 80-7178-926-7

SCHULZ, W. a kol. 1998. Analýza obsahu mediálních sdělení. Praha: Karolinum, ISBN80-7184-548-5

IT-SECURITY

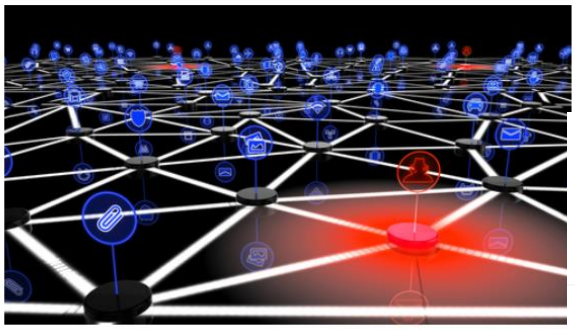
1. Motivation

KrebsonSecurity
In-depth security news and investigation

21 KrebsOnSecurity Hit With Record DDoS

SEP 16

On Tuesday evening, KrebsOnSecurity.com was the target of an extremely large and unusual distributed denial-of-service (DDoS) attack designed to knock the site offline. The attack did not succeed thanks to the hard work of the engineers at **Akamai**, the company that protects my site from such digital sieges. But according to Akamai, it was nearly double the size of the largest attack they'd seen previously, and was among the biggest assaults the Internet has ever witnessed.



The attack began around 8 p.m. ET on Sept. 20, and initial reports put it at approximat
665 Gigabits of traffic per second. Additional analysis on the attack traffic suggests
assault was closer to 620 Gbps in size, but in any case this is many orders of magnitude m
traffic than is typically needed to knock most sites offline.

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUM

RISK ASSESSMENT —

Why the silencing of KrebsOnSecurity opens a troubling chapter for the 'Net

"Free speech in the age of the Internet is not really free," journalist warns.

DAN GOODIN - 9/23/2016, 10:58 PM

DDoS-Attacke

Mit Todessternen auf Spatzen schießen

The banker that ca

By Anton Kivva on September 20, 2016. 10:58 am

MOBILE

BANKING TROJAN | GOOGLE ANDROID | MOBILE BROWSER | MOBILE MALW

CONTENTS

Anton Kivva

Technology

Dropbox hackers stole 68 million passwords - check if you're affected and how to protect yourself

share

The Telegraph

heise Security

« Vorige | Nächste »

Android-Schädling Tordow: Banking-Trojaner mutiert zum Super-Trojaner

28.09.2016 14:00 Uhr – Dennis Schirrmacher

vorlesen

(Bild: Christoph Scholz, CC BY-SA 2.0)

Sicherheitsforscher warnen vor einer Banking-Malware, deren Funktionsumfang sämtliche Träume von Kriminellen erfüllen dürfte: Der Trojaner kann im Grunde alles mit infizierten Android-Geräten anstellen.

'e seen superuser rights exploit adv
privileges is not typical, however, fo
ous other ways that don't require ex
discovered Trojan-Banker.AndroidO
iandy. We had been watching the de
ipabilities had significantly exceeded
ercriminals to carry out new types of

The details of tens of millions of Dropbox users are for sale online CREDIT: DROPBOX

By Cara McGoogan

31 AUGUST 2016 • 11:05AM

heise Security

« Vorige | Nächste »

Tesla Model S lässt sich von fern kapern

20.09.2016 08:24 Uhr – Andreas Wilkens

vorlesen

Screenshot aus dem Demo-Video der Forscher (Bild: Keen Security Lab)

Forscher des chinesischen Internetunternehmens Tencent demonstrieren, wie sie manche Funktionen eines Tesla Model S unautorisiert von fern steuern. Dabei gelang es ihnen auch, ein fahrendes Auto anzuhalten.

Car hacking is the future - and sooner or later you'll be hit

theguardian

Security is finally being taken seriously but the fact that we are increasingly entrusting our lives to self-driving cars creates unease

A Tesla self-driving car on autopilot mode. Researchers explored the potential ways in which such vehicles could be hacked or exploited. Photograph: Bloomberg via Getty Images

Crikey: 43,570,999 user accounts were breached in a hack of Last.fm that occurred in March of 2012, a
2012, Las

DSGVO: Mehr als 1000 US-Portale für Europäer gesperrt

futurezone

08.08.2018

© Bild: Maksim Kabakou - Fotolia / Maksim Kabakou/Fotolia

Noch immer sind mehr als 1000 US-Nachrichtenseiten hierzulande nicht erreichbar, Instapaper ist jedoch wieder verfügbar.

DSG: Verwaltungsstrafe bis EUR 25.000,-- DSGVO: Geldbußen bis EUR 20.000.000,--

https://www.dataprotect.at/info/geldbußen/österreich/

Auch bereits bei **Geltung des Datenschutzgesetzes** (bis 24.5.2018) gibt es **Geldstrafen für Datenschutzverletzungen**; der Strafrahmen ist jedoch (relativ) gering und reicht bis zu **EUR 10.000,--** (für geringe Verstöße) oder bis zu **EUR 25.000,--** (für gröbere Verstöße); es gibt auch **gerichtlich strafbare Handlungen** im DSG (§ 51: Datenverwendung in Gewinn- oder Schädigungsabsicht).

Der **Strafrahmen für Datenschutzverletzungen** steigt mit Geltung der **DSGVO** (25.5.2018) **dramatisch**, und zwar auf 4 % des weltweiten Konzernumsatzes des Vorjahres / EUR 20.000.000,-- als maximale Strafe bzw. 2 % des weltweiten Konzernumsatzes des Vorjahres / EUR 10.000.000,-- bei "geringfügigeren Delikten", wobei dies jeweils die absolute Obergrenze darstellt und immer der Betrag anwendbar ist, der "höher" ist.

dataprotect it-recht

foto: dado ruvic / reuters

Der Facebook-Hack sorgt weiter für Aufregung.

Prozent, das sind maximal 5 Millionen, aus der Europäischen Union. Das teilte die zuständige irische Datenschutzbehörde am Montagabend bei Twitter mit. Facebook habe zugesichert, "bald" ausführlichere Informationen liefern zu können, hieß es in der knappen Stellungnahme weiter.

- Starke Abhängigkeit der modernen Gesellschaft von der IKT (insb. kritische Infrastrukturen).

- IKT wird zur kritischen Informationsinfrastruktur (Critical Information Infrastructure, CII).

- Folgen- machen der IKT len nach [E- de Trends den Schutz zum zentra- Thema (u. a. ckert2008]):


ENTSCHEIDUNGEN ZUM DATENSCHUTZRECHT · 20. September 2018

erste Geldstrafe durch die DSB in Österreich

Die DSB hat die erste **Geldstrafe** verhängt.

EUR 4.800,-- für eine nicht korrekt gekennzeichnete Videoüberwachung, die den öffentlichen Raum mitüberwachte.

https://www.dataprotect.at/2018/09/20/erste-geldstrafe-durch-die-dsb-in-österreich/

Nach Medienberichten (Salzburger Nachrichten, 19.09.2018) wurde gegen einen Betreiber eines Wettlokals in der Steiermark eine Geldstrafe von EUR 4.800,-- verhängt, weil eine **Videoüberwachung nicht ausreichend gekennzeichnet** war und ein **großer Teil des Gehsteigs von der Anlage mitaufgezeichnet** wurde. Die Überwachung des öffentlichen Raums in dieser Art und Weise, nämlich großflächig durch Private ist nicht zulässig.

  - Globalisierung
  - Mobilität (Smartphone, Information at your fingertips)

- Vernetzung
- Ubiquitous/Pervasive Computing, Internet of Things (IoT)
- Industrie x.0
- Smart Home/Grid/Car/*
- Cyber War/Warfare/Espionage/*

- Fazit: Schutz der IKT ist für die Gesellschaft von zentraler Bedeutung!

2. Schutzziele der Informationssicherheit

Informationssicherheit (bzw. IT-Sicherheit) hat das Gewährleisten folgender Schutzziele (Grundwerte) zur Aufgabe [Stallings2006]:

1. Vertraulichkeit (engl. confidentiality)
2. Integrität (engl. integrity)
3. Verfügbarkeit (engl. availability)

C-I-A

4. Authentizität (engl. authenticity)
5. Verbindlichkeit (engl. accountability oder non-repudiation)

2.1. Vertraulichkeit

- **Vertraulichkeit** = engl. Confidentiality
- The protection of data from unauthorized disclosure. [Stallings2006]
- **Traffic Flow Confidentiality** = Schutz vor Verkehrsdatenanalyse
- Vertraulichkeit im täglichen Leben
  - Briefgeheimnis & Fernmeldegeheimnis
  - Amtsverschwiegenheit
  - Beichtgeheimnis
  - Verschwiegenheitspflicht von Rechtsanwälten/Notaren/ Ärzten
  - Geheimhaltungsvereinbarungen/NDAs
- Erreichbar durch Verschlüsselung (Chiffrierung, Encryption)
  - Symmetrische Verschlüsselung
    - ein gemeinsamer geteilter Schlüssel (gleicher Schlüssel für ver- und entschlüsseln) => Schlüsselverteilungsproblem; Strom-/Blockchiffre, Substitution & Transposition
- Asymmetrische Verschlüsselung
  - öffentlicher und privater Schlüssel (öffentlicher Schlüssel für ver- und privater für entschlüsseln) => Schlüsselverteilungsproblem gelöst, algorithmisch aufwändiger und daher langsamer

- Praxis Hybrid: Austausch eines generierten Sitzungsschlüssels über asymmetrische Krypto und dann symmetrische Verschlüsselung

2.2. Integrität

- **Integrität** = engl. Integrity

- The assurance that data received are exactly as sent by an authorized entity (i.e. contain no modification, insertion, deletion, or replay). [Stallings2006]

- Veränderung (z. B. durch Übertragungsfehler oder aktive Angriffe) kann **nicht verhindert** aber **erkannt** werden!

- Authentizität und Integrität werden häufig als Einheit gesehen. Das eine ist ohne das andere nutzlos

- Erreichbar durch den Einsatz von (kryptographischen) Prüfsummen, z. B.
  - CRC (Cyclic Redundancy Check) nur für Übertragungsfehler
  - (Keyed-Hash) Message Authentication Code (MAC, HMAC)
  - Digitale Signaturen

Eine **kryptologische Hashfunktion** oder **kryptographische Hashfunktion** ist eine spezielle Form der Hashfunktion, welche kollisionsresistent oder eine Einwegfunktion (oder beides) ist.

Eine Hashfunktion ist eine Funktion, die eine Zeichenfolge beliebiger Länge auf eine Zeichenfolge mit fester Länge abbildet. Mathematisch ist diese Funktion nicht injektiv (linkseindeutig) und nicht notwendigerweise surjektiv (rechtstotal).

2.3. Verfügbarkeit

- **Verfügbarkeit** = engl. Availability

- Availability is the property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system. [Stallings2006]

- Welche Arten von Performance?
  - Verwendbar bzw. anwendbar
  - Genügend Kapazität
  - Klarer Fortschritt und/oder klar definierte Wartezeiten
  - Vollendung in einem akzeptablen Zeitrahmen
  - ...

- Ansätze zur Realisierung/Garantie von Verfügbarkeit
  - Technisch: Fehlertoleranz und Redundanz durch z. B. RAID, Clustering, ..., gespiegelte Rechenzentren, unterbrechungsfreie Stromversorgung (USV), ...
  - Organisatorisch: Service Level Agreements (SLAs), Verfügbarkeitsklassen

- Verfügbarkeitsklassen
  - 2-6: 99% (Ausfall von ~ 90 Stunden pro Jahr)
  - 99,9%, ... bis 99,9999% (= Ausfall von ~ 30 Sekunden pro Jahr!).

2.4. Authentizität

- Authentizität = engl. Authenticity
  - Entity authenticity= Ich weiß, mit wem ich kommuniziere
  - Data origin authenticity= Ich weiß, von wem die Daten kommen

- Authentifizierung und Authentisierung (authentication)

  - The assurance that the communicating entity is the one that it claims to be. [Stallings2006]

- Authentifizierungsmerkmale = Merkmal mit dem ein Teilnehmer authentifiziert werden kann
  - Basierend auf Wissen (PIN, Passwort)
  - Basierend auf Besitz (Schlüssel, Karte)
  - Basierend auf Eigenschaft (Biometrische Merkmale wie Fingerabdruck, Iris, Stimme, Unterschrift, Retina, ...)
    - Retina = Augenhintergrund (Scan mittels Infrarot)
    - Iris Scan (mittels normaler optischer Kamera)

- Zugangs-/Zugriffskontrolle (Access Control)
  - Authentizität ist Voraussetzung für Zugangs-/Zugriffskontrolle!

2.5. Zugriffskontrolle

- Zugriffskontrolle = engl. access control
- The prevention of unauthorized use of resources. [Stallings2006]
- Die Authentizität der zugreifenden Entität muss sichergestellt sein.
- Es wird geprüft
  - wer Zugriff auf eine Ressource haben darf,
  - unter welchen Bedingungen der Zugriff erfolgen darf und
  - welche Rechte die zugreifende Entität haben darf.
- Prinzip des notwendigen Wissens (Need-to-know principle)
- Grundmodelle
  - Discretionary Access Control (DAC)
    - Festlegung der Rechte ausschließlich auf Basis der Nutzeridentität
- Mandatory Access Control (MAC)
  - Nicht nur Nutzeridentität sondern zusätzliche Regeln und Eigenschaften
- Role Based Access Control (RBAC)
  - Rollenbasierte Rechtvergabe, d. h. nicht auf Basis Nutzeridentität sondern auf Basis von Nutzerrolle (Gruppenzugehörigkeit)

2.6. Verbindlichkeit

- Verbindlichkeit = engl. accountability oder non-repudiation
- Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication. [Stallings2006]
- Verbindlichkeit des Sender = Sender kann den Versand der Nachricht nicht abstreiten.
- Verbindlichkeit des Empfängers = Empfänger kann den Empfang der Nachricht nicht abstreiten.
- Erreichbar durch Einsatz von digitale Signaturen (nicht durch [H]MACs!).

3. Informations- vs. IT-Sicherheit

- Informationssicherheit (Information Security, InfoSec) beschäftigt sich mit Daten jeglicher Form (elektronisch, schriftlich, verbal).
- IT-Sicherheit (IT Security, IT-Sec) konzentriert sich ausschließlich auf die elektronische Datenverarbeitung.
- Definition IT-Sicherheit (übertragbar auf InfoSec, nach [BSI1992]):

> IT-Sicherheit ist der Zustand eines IT-Systems, in dem die **Risiken**, die beim Einsatz dieses IT-Systems aufgrund von **Bedrohungen** vorhanden sind, durch **angemessene Maßnahmen** auf ein **tragbares (akzeptables) Maß** beschränkt sind.

- Höchst wichtige (!!) Konsequenz: Es gibt keine absolute Sicherheit!

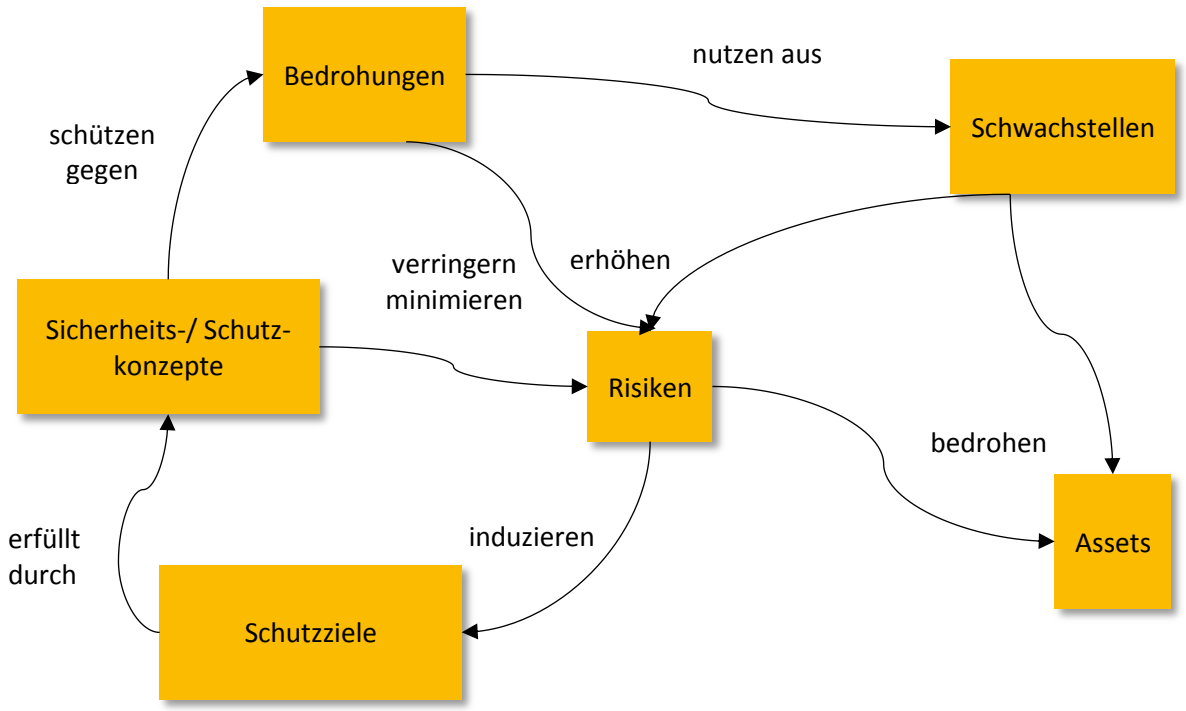
4. Schwachstelle, Bedrohung und Risiko

- Eine Schwachstelle/Verwundbarkeit (eng. weakness/vulnerability) ist eine Schwäche des Systems oder ein Punkt, an dem das System verwundbar sein kann (durch einen Exploit).

- Bedrohungen (eng. threat) ergeben sich aus möglichen Angriffen, die eine oder mehrere Schwachstellen eines Systems ausnutzen, um ein oder mehrere Schutzziele zu gefährden.

- Das Risiko R (eng. risk) einer Bedrohung ist die Wahrscheinlichkeit E des Eintritts eines Schadensereignisses und die Höhe des potentiellen Schadens S, der daraus resultieren kann: R = E • S

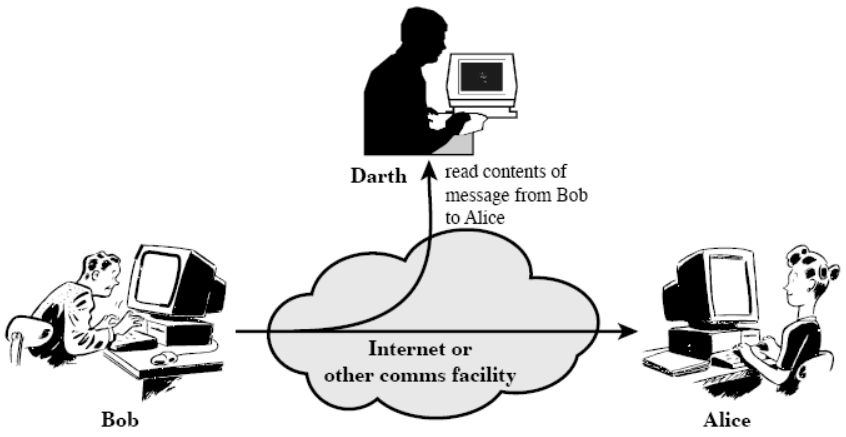
Zusammenhänge



5. Angriffskategorien, -ebenen & Ursachen

Passive Angriffe – Eavesdropping

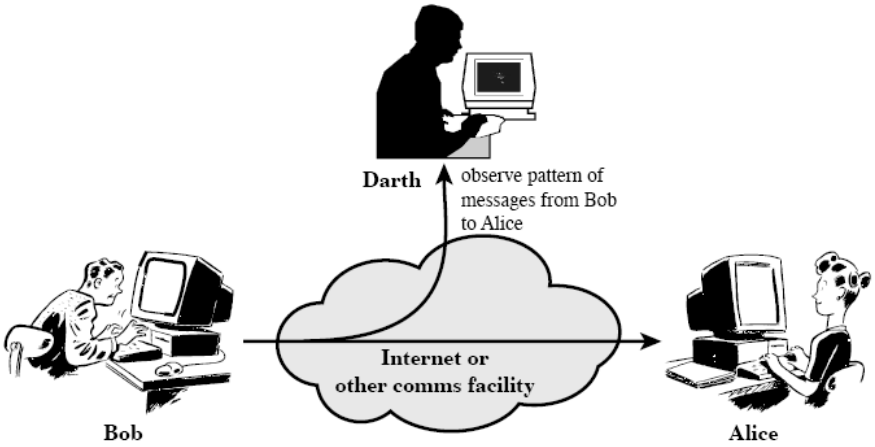
Angreifer hört den Kommunikationskanal ab, greift aber nicht aktiv in die Kommunikation ein.



(a) Release of message contents

Passive Angriffe – Traffic Analysis

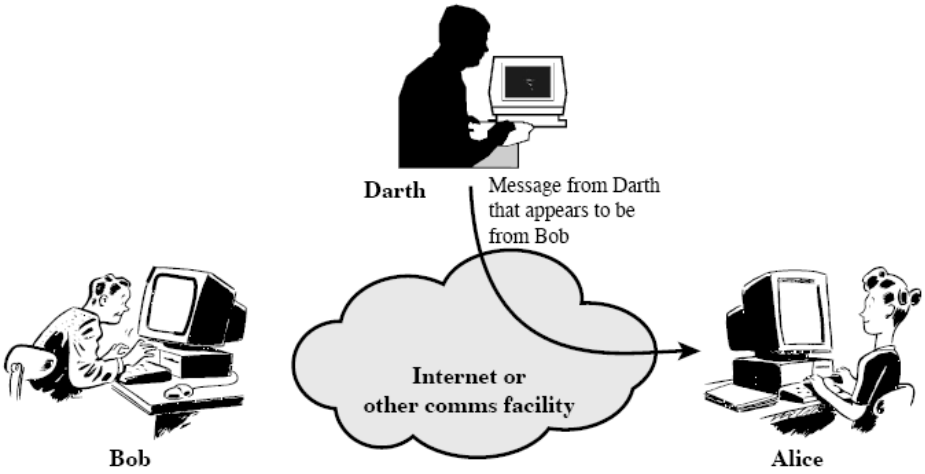
Bei einem verschlüsselten Datenkanal kann ein passiver Angreifer möglicherweise eine Verkehrsdatenanalyse (Traffic Analysis) durchführen (wer kommuniziert wann mit wem?)



(b) Traffic analysis

Aktive Angriffe – Masquerade

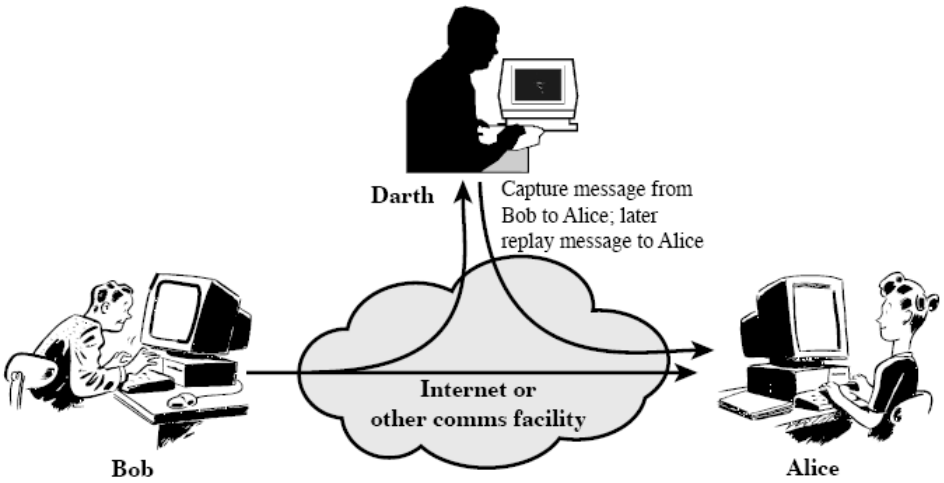
Masquerade: Angreifer gibt sich als jemand anderer aus.



(a) Masquerade

Aktive Angriffe – Insertion & Replay

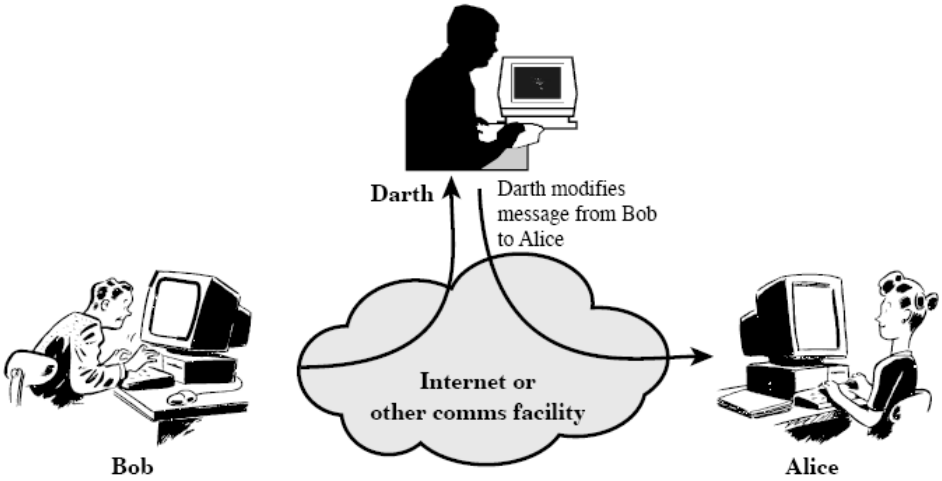
- Insertion: Angreifer fügt Nachrichten(teile) zu einer Kommunikation hinzu.
- Replay: Angreifer sendet aufgezeichnete Daten zu einem späteren Zeitpunkt noch einmal.



(b) Replay

Aktive Angriffe – Modification

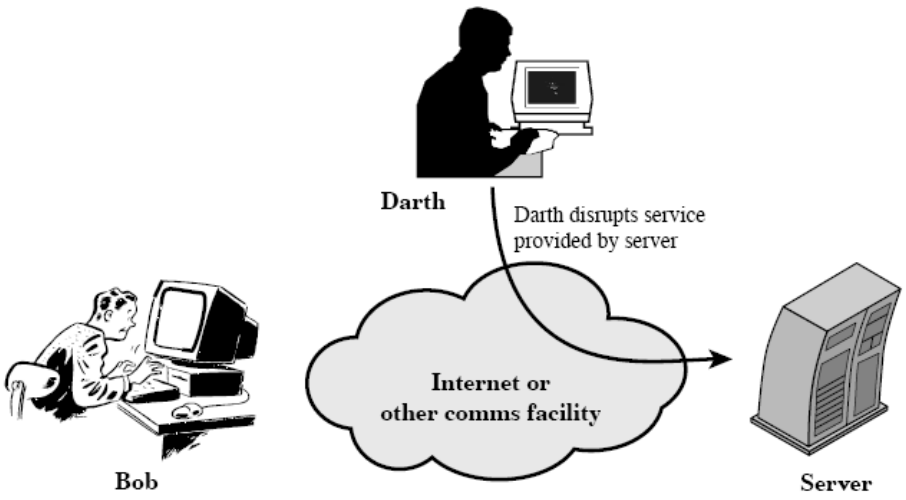
Angreifer verändert eine Kommunikation indem er Nachrichten verzögert, ändert oder löscht.



(c) Modification of messages

Aktive Angriffe – Denial of Service

Angreifer stört die Verfügbarkeit von Kommunikationseinrichtungen



(d) Denial of service

5.1. Angriffsebenen aktuell (Auszug)

- Netzwerk
  - Botnetze
  - (Distributed) Denial of Service (Dienstverweigerung), Reflection, Amplification
  - Spam (Unsolicited Commercial/Bulk E-Mail, Social Media Spam)
  - Man-in-the-Middle-Angriffe (z. B. zum Mitlesen/Modifizieren von Kommunikation)

- Anwendungen (v. a. Web-Anwendungen, s. OWASP Top 10), z. B.
  - Injection (z. B. SQL Injection, Command Injection)
  - Cross-Site Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)
  - Defacements
  - Buffer Overflows

- Benutzer
  - Social Engineering
  - (Spear) Phishing
  - Scareware, Ransomware
  - Spam

5.2. Ursachen

- Fehlende/Mangelnde Identitätsprüfung
  - z. B. schwache Authentifizierungsverfahren, unilaterale Authentifizierung, …

- Fehlende/Mangelnde Input-Validierung
  - z. B. Nutzereingaben in Web-Anwendungen werden server-seitig nicht geprüft

- Psychologische Mängel und Hemmnisse
  - z. B. Social Engineering

- Faktoren Kosten und Zeit in der Produktentwicklung
  - z. B. Kurze Release-Zyklen, fehlende Tests, Sicherheit nicht in Entwicklungs-prozesse integriert

- Organisatorische Mängel
  - z. B. fehlendes Sicherheitsmanagement, fehlende Benutzer-Awareness, …

6. Angreifertypen & ihre Motivation

- Amateure (Script Kiddies)
  - Für die meisten (automatisierten) Angriffe verantwortlich
  - Wenige tiefgehendes Wissen Anwendung von fertigen Angriffstools
  - Motive: Prestige, persönliche Rache, Langeweile

- Crackers vs. Hackers (Blackhats vs. Whitehats)
  - Cracker sind bösartige Hacker (Begrifflichkeiten unklar)
  - Tiefes technisches Verständnis (Studenten, Informatiker)
  - Motive: Prestige, intellektuelle Herausforderung

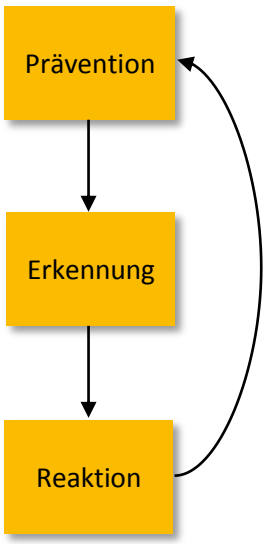
- Kriminelle (Cybercrime)
  - Klassische Kriminelle, die nur aus Profitgründen das Metier wechseln
  - Spamming, Online-Erpressung, Bulletproof-Hosting/-Services, Botnetze (Vermietung), …
  - Gut organisierte Netzwerke mit Verbindungen zum organisierten Verbrechen (z. B. Russian Business Network, Silk Road, …)
  - Motive: Profite

- Terroristen (Cyber-Terrorismus)
  - IT als Angriffsziel Infrastruktur des Zieles schädigen/zerstören
  - Propagandazwecke
  - Kommunikation und Organisation

- Länder und ihre militärischen/nachrichtendienstlichen Apparate (Cyber-War(fare), Cyber-Espionage)
  - Ausschalten der IT-Infrastruktur als Ziel von militärischen Angriffen
  - Wirtschaftliche, politische und militärische Spionage (→ China, USA)
  - Das Internet als Waffe
  - Beispiele: Stuxnet, Flame, Regin, NSA-Überwachungsskandal

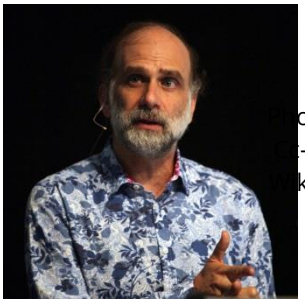
7. Klassifikation von Sicherheitsmaßnahmen

- Maßnahmen zur Verhinderung (a priori)
  - z. B. Authentifizierung, Zugangskontrolle, Einsatz von Verschlüsselung, Firewalls, Härten von Systemen, Sicherheitskonzept, Sicherheitspolitik, Awareness schaffen, …
- Maßnahmen zur Erkennung
  - Dynamisch zur Laufzeit
    z. B. Firewalls, Intrusion Detection Systeme, Log-Analyse, …
- Maßnahmen zur Schadensbegrenzung (a posteriori)
  - z. B. Verschärfen von Kontrollen, Verbesserung von Sicherheitsmaßnahmen, Kappen der Internetverbindung, …



8. Kontinuierliche Ganzheitlichkeit in der Informationssicherheit

Security is not a product; it's process!



Bruce Schneier

Photograph by Rama, CC-by-sa-2.0-fr, from Wikimedia Commons

- Ganzheitlichkeit ist ein kritischer Erfolgsfaktor für IT-/ Informationssicherheit
  - z. B. Firewall zur Filterung von Datenverkehr, aber Benutzer können WLAN Access Points an das Netzwerk anschließen
  - z. B. Zugänge zu den Systemen sind mit Passwörtern geschützt, die am schwarzen Brett angeschlagen sind
  - z. B. Benutzer notieren ihre Windows-Passwörter auf Post-It's auf ihren Monitoren

- IT-/Informationssicherheit erfordert ein Zusammenspiel aus technischen und organisatorischen Maßnahmen!

- IT-/Informationssicherheit im Unternehmen muss vom Management getragen werden!

- IT-/Informationssicherheit muss kontinuierlich gelebt und verbessert werden (Beispiele?)!

8.1. Sicherheitsstandards

- Sicherheitsmanagement/Sicherheitsprozess, z. B.
  - ISO 27000er Familie
  - BSI Grundschutzstandards 100-1, 100-2, 100-3 und 100-4

- Systemsicherheit (Zertifizierung von Produkten), z. B.
  - TCSEC (Trusted Computer System Evaluation Criteria)
  - ITSEC (Information Technology Security Evaluation Criteria)
  - Common Criteria for Information Technology Security Evaluations (ISO 15408)

- Maßnahmenkataloge (technisch/organisatorisch), z. B.
    - BSI Grundschutzkataloge

- Weitere relevante Standards ([IT]-Governance, Compliance), z. B.
    - COBIT (Control Objectives for Information and Related Technology)
    - ITIL (IT Infrastructure Library)

8.2. Rechtsnormen / Gesetze

- Österreichische Strategie zur Cybersicherheit (ÖSCS)
    - Zukünftig: Österreichisches Cybersicherheitsgesetz

- EU Richtlinie über Maßnahmen zur Gewährleistung einer hohen gemeinsamen Netz- und Informationssicherheit (NIS-Richtlinie)
    - Inkorporation in das Österreichische Cybersicherheitsgesetz

- EU Datenschutzgrundverordnung

9. Literatur

[Eckert2008] Eckert, C., Vorlesungsskript zur VO IT-Sicherheit, TU Darmstadt, SS 2008

[Stallings2006] Stallings, W., Cryptography and Network Security, 4th edition, Prentice Hall, 2006

[BSI1992] Bundesamt für Sicherheit in der Informationstechnik (BSI), IT-Sicherheitshandbuch – Handbuch für die sichere Anwendung der Informationstechnik, Version 1.0, März 1992,

[BSI2005] Bundesamt für Sicherheit in der Informationstechnik (BSI), IT-Grundschutzhandbuch, Stand 2005

NETZWERKE

1. Grundbegriffe

In diesem Teil werden all jene Grundbegriffe umfassend erklärt, auf welche Studierende im Bereich der Computernetzwerke stoßen könnten. Es ist an dieser Stelle jedoch wichtig festzuhalten, dass die hier zu findenden Anmerkungen nicht als präzise Definition betrachtet werden können: Sie beziehen sich eher auf ein komplexes Feld mit vielen erwähnenswerten Ausnahmen von der Regel. Nichtsdestotrotz sollen Studierende zumindest eine allgemeine Vorstellung davon haben, was man unter diesen Schlagworten versteht und in der Lage sein, diese effizient zu verwenden.

Client: Darunter versteht man einen Computer, welcher an ein Netzwerk angeschlossen ist. Er hat in der Regel keinen privilegierten Status und seine Möglichkeiten sind jenen von anderen Clients sehr ähnlich. Dieser Begriff begegnet einem auch bei Kennzeichnungssoftware, welche mit dem Server kommuniziert und – auf diese Weise – einen Nutzer mit wesentlichen Diensten (FTP Client, E-Mail-Client usw.) versorgt.

Server: Dieser Begriff bezieht sich auf einen Computer mit einem privilegierten Status im Netzwerk. Von ihm wird verlangt, Clients mit wesentlichen Diensten oder Funktionen zu versorgen. Dazu gehören z.B. die effektive Kommunikation von Clients, die Domainnamen-Überführung in IP-Adressen (DNS), Internetverbindung, Drucken, Filesharing usw. In Beziehung zu einem anderen Server kann ein Server jedoch als Client betrachtet werden. Hierbei handelt es sich in der Regel um speziell konstruierte Computer (spezieller Prozessor, Disk-Matrix usw.).

Router: Dieses Gerät stellt die Paketweiterleitung sicher. Es bindet ein Netzwerkelement, dessen Umgebung es im Detail kennt, mit ein und stellt gleichzeitig sicher, dass die Paketweiterleitung in die richtige Richtung erfolgt. Setzt man ein Paket mit einem Brief gleich, so übernimmt der Router die Rolle des Postdienstes. Doch sein Einsatz reicht noch weiter: Ein Router garantiert qualitativ hochwertige Dienste und stellt TTL sicher (ein Parameter, der die Lebensdauer von Paketen begrenzt, damit sich ein verlorenes Paket nicht für immer im Umlauf befindet). Darüber hinaus sollte er sorgfältig seine Umgebungen erkunden und Netzwerkveränderungen feststellen. Gleichzeitig berechnet er den besten Weg zu weiteren Netzknoten.

Packet: Dies entspricht einem festen Datensatz, welcher über das Netzwerk versendet wird. Daten werden in der Regel nicht (fast nie) als ununterbrochener Informationsfluss gesendet, sondern als Datensätze, sprich Pakete. Dieses aufwändige System macht es sicherer, gewährleistet richtiges Routing sowie die effektive Nutzung der Netzwerkkapazität. Obwohl das Paket eine starre Struktur hat, enthält es in der Regel die Adresse des Senders und Empfängers, Informationen über das Protokoll, Daten oder andere Hilfsin-

formationen.
Netzwerkkarte: Dies ist eine Hardwarekomponente (heutzutage für gewöhnlich fest im Motherboard verbaut), welche die Netzwerkverbindung sicherstellt. Darin enthaltene Informationen werden in Pakete umgewandelt und Pakete in Daten. Die Karte kann ein spezifisches Datentransferprotokoll ausführen (in der Regel Ethernet) und enthält auch einen Kabelanschluss (Twisted-Pair-, Glasfaser-, Koaxial-Kabel). Darüber hinaus enthält die Karte eine spezifische MAC-Adresse, welche weltweit einzigartig ist und zur Identifikation des Computers im Netzwerk dient oder alternativ eine lokale Adresse erzeugt.

Schnittstelle (Port): Diese hat eine Nummer zwischen 0 und 65535, welche eine Anwendung identifiziert, die aktuell als Kommunikationsmittel verwendet wird. Einzelne Anwendungen werden innerhalb des TCP- oder UDP-Protokolls identifiziert. Ports ermöglichen eine Anwendungswartung, stellen (bis zu einem gewissen Grad) die Qualität der Services sicher usw. Es gibt in der Tat feste Anschlussnummern, SMTP hat zB die 25, POP3 die 110, HTTP die 80 oder FTP die 21. Zugleich können sich, auch wenn es unüblich ist, Nutzer oder Anwendungen auf eine andere Port-Nummer verständigen.

Medium: Dieses stellt die physische Umgebung dar, in der die Netzwerkkommunikation ausgeführt wird. Dazu gehören LED-Medien, sprich klassische Kabel (Glasfaser-, Metall-, Twisted-Pair-Kabel) oder Wellenlängen-Medien (Luft, Vakuum), welche Technologien wie WiFi oder GSM nutzen.

Protokoll: Hierbei handelt es sich um eine formelle Sprache, welche die Art der Kommunikation festlegt (IP, TCP, and IMAP4). Es beschreibt genau das Paket, sprich die Geschwindigkeit, mit der es gesendet werden soll. Darüber hinaus definiert es die Sicherheitselemente, korrigiert Fehler usw. Fast jede Schicht des ISO-OSI-Modells enthält seine eigene Gruppe von Protokollen. Diese sind in der Regel an eine Ebene gebunden.

Netzwerkweiche (Switch): Dies ist ein aktives Netzwerkelement, welches Netzwerkteile verbindet. Anders al seine Brücke erlaubt es ein Verteiler, mehr als zwei Netzwerkschleifen miteinander zu verbinden. Kleinere Netzwerkteile haben einen besseren Zugang zum Medium, wodurch eine höhere Übertragungsgeschwindigkeit und Netzwerkauslastung sichergestellt wird.

Repeater: Dieser Begriff bezeichnet ein einfaches Gerät, welches die Signalübertragung über weitere Strecken sicherstellt. Es befindet sich in einem bestimmten Abstand zum Transmitter und verstärkt das ankommende Signal. Dadurch arbeitet es direkt auf der physischen Ebene.

1.1. Wozu braucht man ein Computernetzwerk?

Unternehmen installieren Computernetzwerke zumeist deshalb, weil sie darüber Ressourcen teilen und eine direkte Kommunikation ermöglichen möchten. Zu den Ressour-

cen gehören Daten, Anwendungen und periphere Geräte, zB externe Laufwerke, Drucker oder Modems. Die direkte Kommunikation umfasst das Senden von Nachrichten, das Antworten darauf oder E-Mail-Dienste.

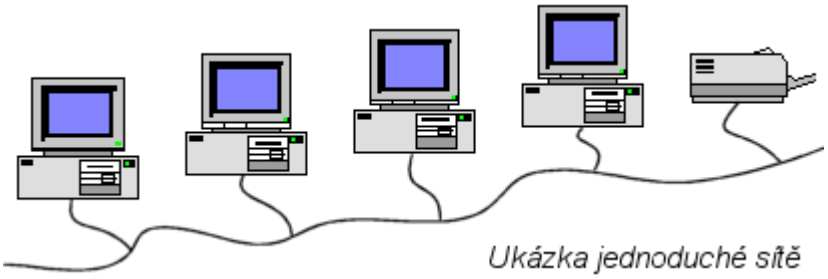


Abb.: Illustration eines einfachen Netzwerks

Computernetzwerk ist ein allgemeiner Begriff für technische Geräte, mit deren Hilfe eine Verbindung und ein Datenaustausch zwischen Computern ermöglicht wird. Darüber hinaus ermöglicht es nutzern, regelkonform zu kommunizieren. Der wichtigste Grund für eine Netzwerkverbindung ist das Teilen von Informationen und technischen Geräten.

Abb.: Eine sternförmige Netzwerkverbindung

Nach ihrer Weitläufigkeit unterscheidet man 4 Computernetzwerke

- PAN (Personal Area Network): Dies ist ein Netzwerk, welches ein sehr kleines Gebiet abdeckt (kleiner als die LAN-Zone). Zum Netzwerk gehören Kommunikationsgeräte wie Handys oder Laptops.

- LAN (Local Area Network): Hierbei handelt es sich um ein lokales Netzwerk, welches ein kleines Gebiet abdeckt, welches zwar größer als die PAN-, aber kleiner als die MAN-Zone ist.

- MAN (Metropolitan Area Network): Dieses Netzwerk ist größer als die LAN-, aber kleiner als die WAN-Zone.

- WAN (Wide Area Network): Dieses Netzwerk deckt ein riesiges Gebiet ab, welches deutlich größer als die MAN-Zone ist.

- VLAN: Darunter versteht man ein virtuelles LAN (Netzwerk). Es handelt sich um ein virtuelles Netzwerk, welches sich innerhalb des LAN-Netzwerks bildet und in einer Hardwarekomponente (Kabel) betrieben wird. VLAN funktioniert auf der Grundlage der Interpretation von Daten, welche über das Netzwerk übertragen werden. Die Daten werden entsprechend des zugehörigen Netzwerks interpretiert.

2. Netzwerktopologie

Die Netzwerktopologie beschreibt die physikalische Verbindung. Genauer gesagt ist es möglich, zwischen einer logischen und einer physikalischen Computerverbindung zu unterscheiden.

Ring: Dieser Begriff bezieht sich auf eine zirkuläre Verbindung von Computern. Hierbei ist ein Computer immer mit zwei anderen verbunden. Der größte Vorteil dieses Modells ist, dass das Netzwerk aufrechterhalten werden kann (vorausgesetzt es ist ein Duplex), wenn ein Element ausfällt. Darüber hinaus kommunizieren die sich am nächsten befindenden Computer (im LAN-Netzwerk) am öftesten miteinander.

Mesh: Hierbei handelt es sich wahrscheinlich um die am seltensten verwendete Topologie. In der Tat gibt es hier keine Grundregeln, weshalb es ad-hoc über sich progressiv verbindende Computer gebildet wird. Auf der anderen Seite ist ihre Beschreibung ziemlich interessant, da sie sehr einfach ist. Logisch betrachtet kann man sagen, dass es sich bei Mesh um ein Internet als Ganzes handelt.

Stern: Dieses Verbindungsmodell ist das gängigste: Seine einzelnen Stationen sind mit einem zentralen Punkt verbunden (zB einem Server), welcher die gesamte Kommunikation übernimmt. Ein großer Vorteil ist sein effektives Management und seine klare Richtung. Der Schwachpunkt ist jedoch der Server.

Baum: Dieser Begriff bezieht sich auf eine andere häufig verwendete Topologie. Hierbei handelt es sich um eine komplexe Computerstruktur, die der Struktur eines mathematischen Baums entspricht. Dadurch können einzelne Computer (sofern es sich nicht um End-zu-End oder Root-Computer handelt) zugleich die Rolle des Servers und des Clients übernehmen.

Bus: Dies ist ein Modell, in welchem die Stationen progressiv an ein Medium angeschlossen werden. Daher hat man es hier mit dem Konzept eines gemeinsamen Mediums zu tun.

Klarerweise gibt es noch eine Reihe weiterer wichtiger Topologien: lineare, wechselseitige und den Extremfall der Punkt-zu-Punkt-Verbindungen. Für gewöhnlich sind Computernetzwerke jedoch irgendwo dazwischen angesiedelt.

2.1. Computernetzwerke und ihre Dienste

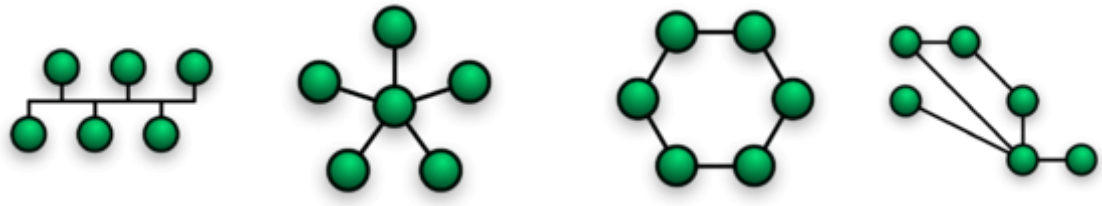
Ein Computernetzwerk ist ein allgemeiner Begriff für technische Geräte, welche eine Verbindung und Datenübertragung ermöglichen. Ebenso erlauben sie Nutzern eine regelkonforme Kommunikation. Der Hauptzweck von Netzwerkverbindungen ist das Tei-

len von Infos und technischen Geräten.

Abb.: Computernetzwerk

2.2. Netzwerktopologien

Der Begriff Topologie bezeichnet die Vernetzung von Computern. Darüber hinaus enthält er Informationen über die spezifische Anordnung der miteinander verbundenen Computer und deren Kommunikation. Mögliche Vernetzungsmethoden sind nachstehend abgebildet:



Bus-Topologie Stern-Topologie Ring-Topologie Unbegrenzte Topologie

Abb.: Topologien: Mögliche Formen von Computernetzwerken

Die Verbindungen der **Bus-Topologie** kommt über ein Übertragungsmedium zustande, welches Bus genannt wird. Mit diesem sind alle Endgeräte, z.B. Netzwerkknoten, verbunden.

Die **Stern-Topologie** bezieht sich auf eine sternförmige Netzwerkverbindung. Es handelt sich hierbei um die häufigste Methode, um Computer zu verbinden. Sie basiert auf einem Computer in der Mitte, welcher auch Zentralcomputer genannt wird. Dieser ist mit den ihn umgebenden Computern vernetzt. Man kann sich vorstellen, dass einer oder mehrere Ende-zu-Ende-Computer durch den Zentralcomputer ersetzt werden. Daher wächst das gesamte Netzwerk progressiv.

Als **Ring-Topologie** bezeichnet man eine Verbindung, bei welcher ein Netzwerkknoten mit zwei anderen Netzwerkknoten in Form eines Rings verknüpft ist. Diese Topologie ist allerdings nicht allzu schnell, da eine große Anzahl an Netzwerkknoten verhindert, dass Daten schnell an den Zielrechner übertragen werden. Darüber hinaus ist in dem Fall, dass einer der Netzwerkknoten nicht funktioniert, ist kein Datentransfer möglich. Somit bricht das gesamte Netzwerk zusammen. Diese Art der Verbindung wird heutzutage kaum noch verwendet, da die Stern-Topologie deutlich zuverlässiger ist. Andererseits sind die Kosten für den Aufbau eines solchen Netzwerks geringer.

3. Was ist das Internet?

Das Internet ist ein globales Netzwerk, welches einem gewöhnlichen Computernetzwerk sehr ähnlich ist. Computer werden miteinander verbunden, wodurch sie in die Lage versetzt werden, miteinander zu kommunizieren und Informationen zu teilen. Das Internet verbindet schon bestehende Netzwerke über eine kohärente Struktur und Distribution miteinander.

Das Internet kann auch als ein Computersystem bezeichnet werden, welches Informationen und Netzwerke enthält, welche uns mit bestimmten Informationen versorgen.

Computer innerhalb des Internets übernehmen entweder die Rolle von Clients oder Servern. Server stellen Internetdienste zur Verfügung, von welchen Clients in der Folge Gebrauch machen. Darüber hinaus stellen Internetdienste auf Anfrage den Datentransfer zu einem Client sicher.

3.1. Die Entstehungsgeschichte des Internets

Während des Kalten Kriegs benötigten die USA ein funktionelles Managementsystem, welches in der Lage sein sollte, die wichtigsten wissenschaftlichen, strategischen und Regierungscomputer (Länder, Militärbasen usw.) miteinander zu verknüpfen. Die Hauptanforderung war es, ein starkes Netzwerk aufzubauen, welches selbst dann, wenn einige Netzwerkknoten ausfielen, weiter aufrecht bleiben und zumindest eine teilweise Kommunikation sicherstellen konnte.

In den 1960er-Jahren wurde die RAND Corporation Company damit beauftragt, eine Lösung zu finden, die es ermöglichte, dass Computer auch nach einem Atomkrieg reibungslos liefen. Die Haupt-aufgabe bestand darin, ein System zu entwerfen, welches selbst nach einer möglichen Zerstörung ihrer Teile stabil funktionieren würde.

Im Jahr 1964 stellte ebendiese Firma eine umfassende Lösung vor, welche auf zwei zentralen Prinzipien beruhte:

- Das Netzwerk enthält keine zentrale Komponente
- Das Netzwerk läuft selbst dann, wenn einige seiner Teile außer Betrieb sind

In Anbetracht dieser Umstände kann man also sagen, dass die US-amerikanische Armee, sprich das Pentagon, das Internet erschaffen hat. Im Jahr 2010 wurden bereits zwei Milliarden aktive Internetnutzer gezählt.

3.2. Die Entwicklung des Internets

Phase Null

Die Phase Null begann, als die US-amerikanische Armee eine effektive Kommunikation zwischen einzelnen Regierungsabteilungen im Falle eines Atomkriegs in der Zeit des Kalten Kriegs aufbauen musste. Das US-Verteidigungsabteilung, sprich die ARPA (Agentur für fortgeschrittene Forschungs-projekte), richtete das ARPANET-Netzwerk ein und schaffte es, dieses Projekt zu finanzieren.

Erste Phase

In der ersten Phase konnte man auf die in späterer Folge mächtigsten Computer zugreifen. Dies waren hauptsächlich jene der Universitäten in den USA. Im Spätherbst des Jahres 1969 wurden die ersten Netzwerkknoten des ARPANET-Netzwerks an den Universitäten in Betrieb genommen. Das Hauptziel war es, nicht nur einen Computer, sondern das gesamte Netzwerk einzubinden. Die Entwicklung fand in allen möglichen Organisationen statt. In der Folge nahmen auch lokale Computer-Netzwerke ihren Betrieb auf. Zu Beginn der 1980er verzeichnete das ARPANET eine rasante Entwicklung, welche auch in anderen Netzwerken stattfand. In der Folge wurden Netzwerke wie Usenet oder BITnet geschaffen. Letztendlich wurden alle Netzwerke an das ARPANET angeschlossen. 1987 waren mehr als 10.000 Netzwerkknoten registriert, zwei Jahre später waren es bereits zehnmal mehr.

In den 80er- und 90er-Jahren wurden Dienste erweitert, mit welchen die Menschen von heute vertraut sind und welche sie nutzen. Diese waren hauptsächlich E-Mail (1971), telnet (1972), TCP (1974) und dessen Updates TCP/IP (1978), DNS (1983) und dessen Start im Jahr 1984. Im Jahr 1980 entschied sich das Pentagon dafür, TCP/IP-Protokolle zu den bevorzugten Protokollen für das Verteidigungsministerium wären. Zwei Jahre später mussten alle Computer, welche an das ARPANET-Netzwerk angeschlossen waren, auf TCP/IP-Protokolle umstellen. Dadurch wurde das ARPANET zu einem Netzwerk und einem Konglomerat aus nebeneinander existierenden und sich neu etablierenden Netzwerken. Allmählich begann sich das Internet abzuzeichnen.

Zweite Phase

Die zweite Phase erstreckt sich über den Zeitraum der Internetentwicklung von 1983 bis 1992. Diese Periode war gekennzeichnet durch ein dramatisches Wachstum des Internets und hauptsächlich durch dessen Verbreitung über die Grenzen Amerikas hinaus. Das ARPANET wurde mit anderen Netzwerken wie dem NFSNET, EUNET, EARN, JUNET usw. verknüpft.

Im November 1983 wurde das DNS Domainsystem eingeführt, welches numerischen Adressen die Zuordnung von Domainnamen erlaubte. Im Jahr 1989 wurde das WWW

(World Wide Web) eingeführt, welches in späterer Folge zu einem integralen Bestandteil des Internets wurde. Im Jahr 1990 wurde das ARPANET stillgelegt und in der Folge eliminiert. Von da an war das NFSNET das zentrale Internetnetzwerk. Im Jahr 1991 wurde die Tschechische Republik an das Internet angeschlossen. 1993 begann man mit der zivilen Nutzung.

3.3. Wozu braucht man das Internet?

Dank des Internets können dessen Nutzer eine Reihe von Diensten verwenden. Diese Dienste werden von Computerprogrammen zur Verfügung gestellt, welche miteinander via Protokoll kommunizieren. Zu den grundlegenden Internetdiensten gehören mitunter:

- **WWW:** Ein System von Webseiten, welches von einem Browser angezeigt wird. Es verwendet HTTP/HTTPS-Protokolle.
- **E-Mail:** elektronische Post
- Versenden von Nachrichten via **SMTP-Protokoll**
- Erhalten von Nachrichten via **POP3- und IMAP-Protokoll**
- **IM (Instant messaging):** Echtzeit-Kommunikation online via ICQ, Jabber, Skype
- **FTP:** Datenübertragung
- **DNS:** Domainnamensystem, welches zum Zwecke besserer Erinnerung eingesetzt wird

Darüber hinaus könnte man eine Reihe weiterer Protokolle erwähnen, zB zum Datenteilen (NFS), Protokolle zum Verbinden zu einem fernsteuerbaren Computer (telnet, SSH, VNC) usw.

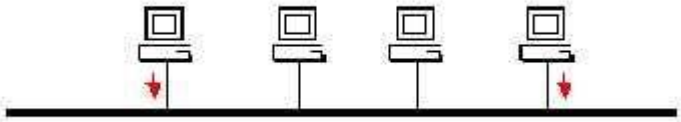
3.4. Ethernet

Ethernet ist eine Technologie für Computernetzwerke. Es repräsentiert ein Set von Technologien, deren Hauptzweck die Datenübertragung in LAN-Netzwerken ist. Die meisten Ethernet-Versionen sind den Standards des IEEE (Institute of Electrical and Electronics Engineers) unterworfen. Ethernet wird über die erste (Bitübertragungs-) und zweite (Sicherungs-) Schicht des ISO/OSI-Referenzmodells implementiert. Darüber hinaus verwendet es hauptsächlich Twisted-Pair- und Glasfasterkabel als Übertragungsmedium: Die ersten Versionen verwendeten hierzu noch Koaxial-Kabel. Der nächste integrale Bestandteil ist der RJ_45 Connector. Es gibt verschiedene Arten von Ethernet, je nachdem, welches Kabel und welche Übertragungsgeschwindigkeit zum Einsatz kommen. Die Übertragungs-geschwindigkeit variiert zwischen 10 Mbit/s und 100 Gbit/s.

Heutzutage ist Ethernet die am häufigsten verwendete Netzwerktechnologie. Diese Technologie basiert auf einem äußerst einfachen Prinzip, welches CSMA/CD genannt

wird – und zwar unabhängig davon, ob es sich um ein gewöhnliches 10 Megabit Ethernet oder eine schnellere Variante davon handelt (Fast und Gigabit Ethernet).
Ein gut strukturiertes Kabelsystem kann diverse Arten von Netzwerktechnologien verwenden, welche auf verschiedenen Arten von Übertragungsmethoden basieren, z.B. Ethernet, Token Ring, CDDI, ATM usw.

Bei **CSMA** (Carrier Sense Multiple Access) handelt es sich um eine Station, welche bereit ist, Daten und zu übertragen und "überprüft", ob das Übertragungsmedium (Kabel) von einer anderen Station verwendet wird. In so einem Fall versucht die Station, später darauf zuzugreifen, bis das Medium frei ist. Ab dem Moment, ab dem das Medium frei ist, beginnt die Station mit der Datenübertragung.



CD (Collision Detection) bezieht sich auf eine Station, welche während der Übertragung überwacht, ob das Medium ein Signal anzeigt, welches mit den Übertragungsebenen übereinstimmt (sprich, um nicht Signal 1 zu übertragen, wenn gerade Signal 0 übertragen wird). Diese Art der komplexen Signal-Interaktion wird Collision genannt. Im Falle einer **Detection** sendet die Collision-Station ein JAM-Signal aus und beide (alle) Stationen erzeugen zum selben Zeitpunkt der Übertragung einen zufälligen Zeitwert, nach welchem sie versuchen werden, die Übertragung zu wiederholen.

Tatsächlich war diese Effektivität verantwortlich dafür, dass die Preise von AC-Adaptern und aktiven Elementen fielen und sich Ethernet in der Folge merklich verbreitete. Gleichzeitig brachte die Effektivität einer solchen Lösung einen gravierenden Nachteil mit sich: Durch die steigende Zahl der Netzwerkknoten erhöhte sich gleichzeitig die Zahl der Collisions. Dadurch verringert sich die theoretische Durchsatzleistung des Netzwerks. Das Knoten-Set, durch welches die gewöhnliche Aktivität eine Collision auslösen kann, wird Collision Domain genannt. Logisch kann daraus abgeleitet werden, dass die **Collision Domain** so klein wie möglich sein sollte. Angelegte aktive Elemente haben eine unterschiedliche Beziehung zur Collision Domain: Auf der einen Seite erweitern mache davon die Collision Domain, auf der anderen Seite trennen manche diese ab. Deshalb kann man durch sorgfältige Auswahl der aktiven Elemente die Durchsatzleistung des Netzwerks steuern.

Zusätzlich wurde, abgesehen von der Collision Domain, der Begriff **Broadcast Domain** eingeführt. Das Computernetzwerk enthält zwei Arten von Paketen: Unicasts und Non-Unicasts. Der Begriff Unicasts bezieht sich auf Pakete mit einem bestimmten Adressaten, welcher von der Netzwerkadresse zur Verfügung gestellt wird. Andererseits neigen Non-Unicasts dazu, eine Gruppenadresse zu verwenden und sind entweder für alle

Netzwerk-nutzer (Broadcasts) oder eine ausgewählte Nutzergruppe (Multicasts). Das Hauptproblem ist, dass ein Computer sich mit Non-Unicast beschäftigten muss, auch wenn er darauf nicht ausgelegt ist. Deshalb erhöht sich gleichzeitig mit der Zahl der Netzwerkknoten durch eine große Zahl an Non-Unicasts auch die Broadcast Domain. Aus diesem Grund ist es notwendig, die Größe der Broadcast Domain in vernünftigen Grenzen zu halten. Angelegte aktive Elemente haben eine unterschiedliche Beziehung zur Broadcast Domain. Folglich kann durch deren sorgfältige Auswahl die Durchsatzleistung des Netzwerks kontrolliert werden.

Paketformat

Wie zuvor beschrieben wurde, werden alle Ethernet-Geschwindigkeitsmodifikationen mit derselben Kommunikationsmethode CSMA/CD erreicht. Sie nutzen allerdings auch dieselbe Paketgröße und dasselbe Paketformat. Das Ethernetpaket ist auf der ersten und zweiten OSI-Schicht definiert.

4. Standardisierung v. Computernetzwerken, TCP/IP Stack

4.1. Standardisierung von Computernetzwerken

Zur Kernstrategie führender Hersteller gehörte das Ausarbeiten und Warten von Spezifikationen, welche genau mit den Produkten der Firmenkunden kompatibel sind. Das größte Manko dieser Computernetzwerke, welche als homogene oder geschlossene Systeme bezeichnet wurden, bestand darin, sich sowohl an die Hersteller von technischer Geräte als auch die Hersteller von Software anzupassen. An dieser Stelle werden ein paar firmeneigene Protokolle genannt: DECnet von DEC, SNA (Systemnetzwerkarchitektur von IBM), SPX/IPX (Sequenzierter/Internet-Paketaustausch von Novell). Im Gegensatz zu diesen geschlossenen Systemen beziehen sich offene oder heterogene Systeme auf Computersysteme, welche in Einklang mit bewährten internationalen Standards entwickelt wurden und von mehreren unabhängigen Produzenten gekauft werden können. Die Umsetzung offener Systeme erlaubt den Programmen, welche in lokalen oder ferngesteuerten Netzwerksystemen zum Einsatz kommen, eng zusammenzuarbeiten und ermöglicht eine einheitliche Kommunikation zwischen den Benutzern der Anwendungen.

Der Begriff OSI (Open Systems Interconnection) betrifft die technische Ausstattung und Computernetzwerksoftware. Im Bereich der Computernetzwerke sind die internationalen Organisationen CCITT (Comité Consultatif International de Télégraphique et Téléphonique als eine der drei zentralen Komitees der International Telecommunications Union ITU) und ISO (International Organzisation for Standardization) die wichtigsten. Sie beschäftigten sich damit, geeignete Standards einzuführen. In der Tat sind Regierungsinstitute oder Organisationen, welche in den einzelnen Mitgliedsländern für die Telekommunikation verantwortlich sind, die wichtigsten Mitglieder der CCITT. Die CCITT gibt detaillierte Empfehlungen heraus, welche nach einer grundsätzlichen Genehmigung von der ITU zu internationalen Standards werden. Im Gegensatz zur CCITT ist die ISO eine freiwillige nichtstaatliche Organisation, dessen Mitglieder Standardisierungsinstitute sind. Eine weitere wichtige Organisation ist die IEEE, bei welcher es sich um eine prestigeträchtige amerikanische Vereinigung von Elektro- und Elektronik-Ingenieuren handelt. Diese Vereinigung gibt Zeitschriften heraus, halt Konferenzen ab und bildet eine professionelle Körperschaft, welche professionelle Standards einführt. Die Standardgruppe IEEE 802 ist im Bereich der lokalen Computernetzwerke erwähnenswert. Im Jahr 1979 übernahm die ISO einen Standard namens Reference Model of Open Systems Interconnection (RM OSI).

Netzwerkkommunikation

Zu den Hauptaufgaben eines Netzwerks gehört die Datenübertragung von einem Computer zu einem anderen. Dieses komplexe Unterfangen kann in die folgenden Schritte unterteilt werden:

- Datenüberprüfung
- Aufteilen der Daten in nützliche Datenbanken
- Versorgen jeder Datenbank mit nützlicher Information (Quelle und Ziel)
- Bereitstellen relevanter Informationen über zeitliche Abstimmung und Fehlersuche
- Übermitteln von Daten an ein Netzwerk und Versenden dieser an einen geeigneten Ort

4.2. Grundbegriffe RM OSI

Eine **Schicht** ist durch ihre nutzbaren Funktionen genau definiert. Jede Schicht, mit Ausnahme der obersten und untersten, verbindet die jeweils darunter- und darüberliegende Schicht über eine Schnittstelle miteinander. Die oberste Schicht bindet den Umsetzungsprozess über eine Schnittstelle an. Die unterste Schicht bindet zudem über eine Schnittstelle das physikalische Medium an.

Eine Entität ist ein Objekt, welches effektiv in einer bestimmten Schicht arbeitet. Der Begriff bezieht sich für gewöhnlich auf eine Programmgruppe. Zu den unteren Ebenen gehört die Hardware (I/O Port). Der unmittelbare Bereitsteller bzw. Nutzer eines Dienstes beziehen sich nicht auf eine Schicht, sondern auf eine bestimmte Entität einer Schicht.

Zu einem **Protokoll** gehören Entitäten in denselben Schichten verschiedener offener Systeme. Sie **kommunizieren** miteinander.
Der Begriff Service bezieht sich auf Entitäten einer bestimmten Schicht, welche dessen Funktion ausführen. Darüber hinaus stellen sie die höhere Schicht mit den geeigneten Diensten zur Verfügung. Um diese Funktionen ausführen zu können, verwenden sie Dienste aus der direkt darunter befindlichen Schicht. Einzelne Dienste werden von Service Access Points (SAP) angeboten, welche über ihre Adressen identifiziert werden. Diese Dienste lassen sich wie folgt einteilen:

- **Verbindungsorientierte Services:** Zwei Entitäten auf denselben Ebenen müssen zuerst eine Verbindung herstellen, bevor eine direkte Kommunikation gestartet werden kann. Nach dem erfolgreichen Zustandekommen einer Verbindung verschickt der Sender eine Nachricht und der Empfänger erhält diese. Dieser Service ist vergleichbar mit einer Telefonverbindung.

- **Verbindungslose Dienste** setzen kein Zustandekommen einer starken Verbin-

dung zwischen einem Sender und einem Empfänger voraus. Stattdessen betrachten sie jedes Einzelteil der übertragenen Daten für sich als Ganzes, welcher mit der Adresse des letzten Empfängers ausgestattet ist und unabhängig von anderen Nachrichten zugestellt wird. Einzelne Nachrichten können auch auf andere Wege übermittelt werden.

- **Zuverlässige Dienste** stellen Services zur Verfügung, welche niemals irgendwelche Daten verlieren. Dazu gehören in der Regel Dienste, welche von einem effektiven Mechanismus für Bestätigungsmeldungen aufrechterhalten werden.

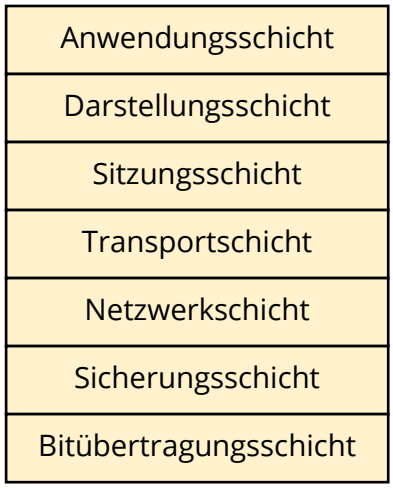
- **Unzuverlässige Dienste** stellen keine Bestätigungsmeldungen zur Verfügung, bieten jedoch Services mit hoher Zuverlässigkeit.

4.3. ISO/OSI Referenzmodell

Dieses beliebte Modell wurde von der ISO als einheitlicher Standard für miteinander verbundene Systeme verschiedener Arten und Konzepte entworfen, welche von verschiedenen Herstellern definiert wurden. Das erstellte Modell enthält sieben Schichten. Diese sieben Schichten schaffen eine komplexe Hierarchie, welche mit Anwendungen an der Spitze beginnen und mit physikalischen Verbindungen auf der untersten Ebene enden. Das OSI Referenzmodell enthält zwei Modelle der Kommunikation:

- Horizontal: protokollbasiertes Modell, über welches Programme und Prozesse verschiedener Computer kommunizieren,
- Vertikal: servicebasiertes Modell, über welches die Schichten eines einzelnen Computers miteinander kommunizieren.

Tatsächlich erfordert eine effektive Kommunikation folgende Elemente: Man benötigt zwei Seiten, welche miteinander kommunizieren möchten sowie eine gemeinsame Sprache (Protokoll), mit deren Hilfe die Seiten miteinander kommunizieren können. Darüber hinaus kommunizieren die vertikalen Schichten via API (Application Program Interface).



4.3.1. ANWENDUNGSSCHICHT

Diese Schicht legt genau fest, in welcher Umgebung Netzwerkanwendungen mit dem Netzwerkdienst kommunizieren. Der Endbenutzer verwendet das Netzwerk, um die Anwendung auszuführen, Daten zu übertragen, Nachrichten zu senden, sich aus der Ferne einzuloggen usw. Die Anwendungsschicht-protokolle enthalten vor allem Anwendungsprogramme, aber auch Netzwerküberbauten, welche es der Station ermöglichen, sich mit dem Netzwerk zu verbinden. Programme und Protokolle, welche Dienste der Anwendungsschicht bereitstellen, sind mitunter:

- NICE (Network Information and Control Exchange), welches die Überwachung und Administration des Netzwerks sicherstellt.
- FTAM (File Transfer Access and Management), welches für die Dateienverwaltung aus der Ferne zuständig ist.
- X.400, welches Protokolle und Funktionen zum Versenden von Nachrichten und elektronischer Post festlegt.
- CMIP, welches für die Netzwerkadministration verantwortlich ist, die auf den von OSI definierten Rahmenbedingungen basiert.
- Telnet, welches eine Terminalemulation und Fernverbindung bereitstellt.
- Rlogin, welches eine UNIX-Umgebung mit Fernverbindung bereitstellt.

4.3.2. DARSTELLUNGSSCHICHT

Diese Schicht gibt der Datenübertragung ihre Form. Die Daten, welche über das Netzwerk übermittelt werden, können entweder Texte, Ziffern oder allgemeine Datenstrukturen sein. Darüber hinaus enthält diese Schicht Spezifikationen für das Kodieren und Dekodieren von Zeichensätzen. Eine Datenkomprimierung und Chiffrierung kann auch in dieser Schicht stattfinden. Darüber hinaus stellt die Darstellungsschicht der direkt darüber befindlichen Anwendungsschicht Services zur Verfügung und nutzt eine Sitzungsschicht, die sich unmittelbar darunter befindet.

4.3.3. SITZUNGSSCHICHT

Auf dieser Ebene werden Prozesse angestoßen, welche den Datentransfer steuern, deckt Übermittlungs- und Übertragungsfehler fest und legt Aufzeichnungen über Rundfunkübertragungen an. Darüber hinaus unterbricht diese Schicht Sitzungen zwischen den beiden Endteilnehmern (dem Nutzer und dem Zielcomputer) bzw. hält diese aufrecht. Um eine Sitzung zu starten, bedarf es einer über die Transportschicht hergestellten Verbindung, mittels welcher eine Kommunikation zwischen den beiden Sitzungsteilnehmern ermöglicht wird. Ebenso entscheidet sie, wer Signale übermitteln kann oder unterbricht die bestehende Sitzung. Zu den Sitzungsschichtprotokollen gehören:

- ADSP (AppleTalk Data Stream Protocol): Dieses erlaubt zwei Netzwerkknoten, eine prekäre Verbindung für den Datentransfer herzustellen.
- NetBEUI bezieht sich auf die Umsetzung und Entwicklung von NetBIOS.
- NetBIOS verwendet die 5., 6. und 7. Schicht und bietet Dienste zur Sitzungsüberwachung.
- PAP (Printer Access Protocol) ermöglicht den Zugriff auf PostScript-Drucker im AppleTalk-Netzwerk.

4.3.4. TRANSPORTSCHICHT

Diese Schicht ermöglicht die Funktionstüchtigkeit eines geeigneten Verbindungsaufbaus und splittet Daten in kleinere Teile, sprich Pakete auf, zu welchen die übertragenen Daten gebündelt werden. Nach deren Empfang werden die Daten aus dem Paket entnommen und in ihre ursprüngliche Form gebracht. Dadurch stellt diese Schicht sicher, dass beliebig große Nachrichten übertragen werden, obwohl ihre einzelnen Pakete eine begrenzte Größe haben. In der Tat ist die Transportschicht essentiell, da sie sich zwischen oberen und unteren Schichten befinden, welche netzwerkorientiert sind.

4.3.5. NETZWERKSCHICHT

Diese Schicht beschreibt Prozesse, welche Daten zwischen Netzwerkadressen versenden und überprüfen, ob die gesamten Nachrichten rechtzeitig gesendet wurden. Darüber hinaus stellt sie einangenehmes Versenden von Paketen sicher. Die Netzwerkschicht muss überdies die besondere Netzwerktopologie kennen, sprich den Weg der direkten Verbindung der einzelnen Netzwerkknoten untereinander.

4.3.6. SICHERUNGSSCHICHT

In dieser Schicht, welche auch Verbindungsschicht genannt wird, finden jene Verfahren statt, welche zum Aufspüren und Korrigieren von Fehlern auf der Datenebene während der Übertragung von Daten zwischen der Bitübertragungsschicht und den darüber liegenden Schichten dienen. Die Sicherungsschicht erstellt Pakete relevanter Netzwerkarchitektur. Trotzdem kommt es auf dem Übertragungsweg immer wieder zu Fehlern, welche zur Folge haben, dass die erhaltenen Bites sich von den versendeten unterscheiden. Die Bitübertragungsschicht kümmert sich nicht um die Relevanz einzelner Bits. Diese Art von Fehlern wird in der Sicherungsschicht aufgespürt. Diese überprüft, ob die ganzen Pakete (Verschicken unterschiedlicher Prüfsummen usw.) korrekt übertragen wurden. Darüber hinaus übermittelt die Sicherungsschicht dem Sender unmittelbar eine Empfangsbestätigung der einwandfrei übertragenen Pakete, wohingegen ein Fehler deren wiederholte Übermittlung erfordert.

4.3.7. BITÜBERTRAGUNGSSCHICHT

Diese Schicht erhebt die elektrischen, mechanischen und funktionalen Anforderungen, welche an die Verarbeitung von Netzwerkdaten gestellt werden. Im Großen und Ganzen scheint die Aufgabe einfach zu sein: Die Schicht stellt die Übertragung einzelner Bits zwischen einem Empfänger und einem Sender sicher.

4.4. TCP/IP Modell

TCP/IP steht für Transmission Control Protocol/Internet Protocol. Es bezieht sich auf eine Standard-Protokolldatei, welche zB im Internetnetzwerk verwendet wird, welches zum größten Netzwerk der Welt geworden ist.

Wie die Geschichte zeigt, hat die DARPA (Defence Advanced Project Agency) 1969 ein Forschungs- und Entwicklungsprojekt gestartet, welches darauf abzielte, ein experimentelles Netzwerk mit Datenpaketvermittlung aufzubauen. Dieses Netzwerk wurde ARPANET genannt und wurde entworfen, um Methoden zu studieren, welche eine verlässliche und anbieterunabhängige Daten-kommunikation erlauben. Alles in allem war das Experiment sehr erfolgreich und eine große Zahl von Organisationen begann damit, dessen Dienste für die Alltagskommunikation zu verwenden. Im Jahr 1975 wurde das ARPANET von einem experimentellen in ein operierendes Netzwerk umgewandelt. Das Fundament für das TCP/IP-Protokoll wurde gelegt, als das ARPANET bereits ein operierendes Netzwerk war. Dieses Protokoll wurde 1983 als Militärstandard übernommen und ungefähr von diesem Zeitpunkt an verbreitete sich der Begriff Internet über den gesamten Globus.

Vorteile von TCP/IP

- Offener Protokollstandard, welcher frei verfügbar ist und unabhängig von einer bestimmten technischen Ausstattung oder eines Betriebssystems entwickelt wurde.
- Vollkommene Unabhängigkeit von einer bestimmten physikalischen Netzwerkhardware, wodurch es möglich ist, TCP/IP innerhalb einer großen Anzahl von Netzwerken zu betreiben. So kann TCP/IP mitunter über Ethernet, Token Ring, Telefonleitungen und praktisch jedes physikalischen Übertragungsmedium laufen.
- Eine allgemeine Regelung erlaubt es jedem TCP/IP-Gerät, jedes andere Gerät im gesamten Netzwerk anzusprechen.
- Standardisierte High-Level-Protokolle, welche allgemein verfügbare Nutzerdienste anbieten.

4.5. Architektur von TCP/IP-Protokollen

Protokolle entwerfen formale Verhaltensrichtlinien innerhalb des Datenverkehrs. In homogenen Netzwerken wird dieses Regelwerk vom Systemanbieter formuliert. TCP/IP erstellt ein heterogenes Netzwerk mit offenen Protokollen, welche nicht von den Unterschieden zwischen Betriebssystemen und Architekturen abhängig sind. Sie sind für jeden verfügbar und ihre Entwicklung und Anpassungen unterliegen den Vertragsbedingungen. Die genaueste Information über TCP/IP findet sich im RFC (Request for Comments): Diese Dokumente enthalten die letzten Aktualisierungen aller Standardspezifikationen.

TCP/IP wird gemeinhin al sein Modell betrachtet, welches aus weniger Schichten als das OSI-Modell besteht. Die meisten der TCP/IP-Beschreibungen definierten die Protokollarchitektur mit drei bis fünf Ebenen.

Netzwerkschnittstellenschicht

Diese Schicht kümmert sich um alles, was man mit der Steuerung eines bestimmten Übermittlungs-pfades, der direkten Übertragung und dem Empfang von Datenpaketen verbindet. Darüber hinaus bedarf sie einer bestimmten Übermittlungstechnologie (sprich alle Industriestandards wie Ethernet, IEEE 802.x und FDDI).

Internetschicht

Diese Schicht wird manchmal auch IP-Schicht genannt, in Anlehnung an das IP-Protokoll. Sie stellt sicher, dass einzelne Pakete vom Sender zu ihrem tatsächlichen Empfänger geschickt werden – unabhängig davon, ob zwischen ihnen eine direkte Verbindung besteht. Unter Berücksichtigung des nicht-verbundenen Übertragungscharakters (IP-Protokoll) wird ein elementarer Datagrammdienst auf der Ebene dieser Schicht bereitgestellt.

Das Internetprotokoll ist ein Eckpfeiler des Internets. Zu seinen Funktionen gehören:

- die Datagrammdefinition, welche die grundlegende Übermittlungseinheit ist,
- die Definition des Internetadressschemas,
- die Datenübertragung zwischen der Netzwerk- und Transportschicht,
- das Hinführen von Datagrammen zu entfernten Zielen,
- das Bereitstellen einer Fragmentierung und eines Datagrammaufbaus.

Datagramm bezieht sich auf ein Paketformat (Paket ist ein Datenblock, welcher Informationen enthält, die für dessen Zustellung notwendig sind), welches vom Internetprotokoll definiert ist.

Transportschicht

Diese Schicht wird gemäß des Protokolls auch TCP-Schicht genannt. Sie ist dafür zustän-

dig, die Übermittlung zwischen den Endteilnehmern sicherzustellen, welche im Fall einer TCP/IP-Architektur Anwendungsprogramme sind. Je nach deren Anforderungen kann die Transportschicht den Daten-fluss in beide Richtungen regulieren, die Übermittlungszuverlässigkeit sicherstellen und auch den nicht-verbundenen Charakter der Übertragung innerhalb der Netzwerkschicht in einen verbundenen Charakter ändern.

Anwendungsschicht

Die Anwendungsschicht ist die oberste Schicht der TCP/IP-Architektur: Ihre Entitäten repräsentieren individuelle Anwendungsprogramme, welche – im Gegensatz zu RM OSI – direct mit der Transport-schicht kommunizieren. Mögliche Darstellungs- und relationale Dienste werden von den Anwendungsprogrammen selbst bereitgestellt.

4.6. Architektur kommunizierender Systeme

Begriffsbestimmungen

- System
  - ein unabhängiges Ganzes, welches in der Lage ist, Prozesse und Übertragungsinformationen anzuregen

- Netzwerkarchitektur
  - System aus Schichten, Diensten, Funktionen und Protokollen
  - stimmt mit der Struktur der Netzwerkausstattung überein

- Offene Architektur
  - entsprechende Standards, welche die Architektur beschreiben, sind öffentlich zugänglich
  - alle Systeme, welche die Standards erfüllen, sind miteinander verknüpfbar

- Schichtprotokoll
  - Kooperationsregeln der Entitäten derselben Schicht und anderer Systeme

- Schnittstellenprotokoll
  - SW-Schnittstelle
  - Kooperationsregeln von benachbarten Schichten
  - Dienstelemente werden verwendet
  - Kommunikation über Service Access Points (SAP)

5. ISO/OSI Modell, ausgewählten Protokolle

5.1. Computernetzwerke - ISO/OSI-Modell

Das ISO/OSI-Modell bezeichnet ein Kommunikationsmodell, welches “International Standards Organization/Open System Interconnection“ genannt wird. Es bezieht sich auf ein empfohlenes Modell, welches von der ISO im Jahr 1983 definiert wurde. Dieses teilt die übliche Kommunikation zwischen Computern in sieben miteinander verbundene Schichten auf. Die entsprechenden Schichten werden als Protokollschichtensatz bezeichnet.

Die Hauptaufgabe jeder Schicht ist es, der jeweils darüber befindlichen Schicht bestimmte Services bereitzustellen und eben jene übergeordnete Schicht nicht mit Details über die bestimmte Service-bereitstellung zu belasten. Bevor Daten von einer Schicht in die andere übertragen werden, werden diese in Pakete unterteilt. In der Folge werden alle Pakete in jeder Schicht mit zusätzlichen Informationen (Format, Adresse) angereichert, welche für eine erfolgreiche Übertragung über das Netzwerk notwendig sind.

Das vorgeschlagene Modell zeigt die folgenden Schichten (jede übergeordnete Schicht nutzt Funktionen der darunter befindlichen Schicht).

Eine kurze Beschreibung der einzelnen Schichten: Die einzelnen Schichten werden in derselben Reihenfolge erklärt, wie sie in der Grafik nummeriert sind.

1. Bitübertragungsschicht

Diese definiert Kommunikationsmittel mit einem mobile Medium und die technischen Mittel der Schnittstelle. Darüber hinaus definiert sie die physikalischen, elektrischen, mechanischen und funktionellen Parameter, welche sich auf die physische Verbindung

der einzelnen Komponenten beziehen. Das heißt, sie befasst sich mit der Hardware.

2. Sicherungsschicht

Diese hält die Integrität des Datenflusses von einem Netzwerkknoten zu einem anderen aufrecht. Diese Aktivität umfasst auch die Datenblocksynchronisierung und deren Flussteuerung. Diese Schicht befasst sich folglich auch mit der Hardware.

3. Netzwerkschicht

Diese legt Protokolle für Datenrichtungen fest, über welche der Informationstransfer zum erforderlichen Netzwerkknoten sichergestellt wird. Dies bedeutet, dass die richtige Richtung – außer, man entscheidet sich dafür – nicht im lokalen Netzwerk sein muss. Diese Schicht bezieht sich auch auf die Hardware: Wenn sich der PC jedoch mit der Richtung zwischen zwei Netzwerkkarten befasst, betrifft sie die Software.

4. Transportschicht

Diese definiert die Protokolle für strukturierte Nachrichten und stellt deren einwandfreien Transfer sicher (übernimmt ein paar Fehlerüberprüfungen). Darüber hinaus stellt sie die Partitionierung in Pakete und die Bestätigung sicher. Diese Schicht befindet sich auf der Software-Ebene.

5. Sitzungsschicht

Diese koordiniert die Kommunikation und hält Sitzungen so lange wie notwendig aufrecht. Ebenso übernimmt sie Sicherheits-, Anmeldungs- und Verwaltungsfunktionen. Sie bezieht sich auf Software.

6. Darstellungsschicht

Diese erkundet den Art und Weise der Datenformatierung, -darstellung, -umwandlung und -kodierung. Das heißt, sie beschäftigt sich mit Umlauten, Interpunktion, (De-)Komprimierung und Datenverschlüsselung. Sie befindet sich ebenfalls auf der Software-Ebene.

7. Anwendungsschicht

Hierbei handelt es sich um die oberste Schicht. Sie erkundet die Art und Weise, auf bzw. über die Netzwerke mit Anwendungen kommunizieren können, zB Datenbanksysteme, elektronische Post oder Programme für Terminalemulationen. Sie nutzt Dienste darunter befindlicher Schichten, wo-durch sie von Problemen technischer Netzwerkressourcen abgegrenzt wird. Sie ist softwarebasiert.

5.2. ISO/OSI-Referenzmodell: Sieben Schichten

Die Architekten des ISO/OSI-Referenzmodells kamen überein, dass eine Netzwerksoftware idealerweise sieben Schichten haben sollte. Es bleibt jedoch noch die Frage zu klären, mit welchen Schichten bzw. welchen damit verbundenen Aufgaben man sich auseinandersetzen muss. Auf eben diese wird nachfolgend in hierarchisch aufsteigender Reihenfolge eingegangen:

1. Bitübertragungsschicht

Ihre Hauptaufgabe scheint recht einfach zu sein: Eine Übertragung von einzelnen Bits zwischen dem Empfänger und dem Sender über einen physikalischen Transferpfad zu gewährleisten, welcher sich unter vollständiger Kontrolle dieser Schicht befindet. Nichtsdestotrotz muss man sich mit einer Reihe von Problemen technischer Natur beschäftigen, zB dem Spannungsniveau der Logik 1 und der Logik 0, wie lange ein Bit „währt", wie viele Kontakte und welche Form Kabelanschlüsse haben sollten, welche Signale über diese Kabel übermittelt werden, welche Relevanz diese haben, deren Zeitverlauf usw. Deshalb fallen diese Angelegenheiten eher in den Kompetenzbereich der Elektroingenieure und Techniker.

2. Sicherungsschicht

Die Bitübertragungsschicht stellt in der Regel Mittel zum individuellen Bittransfer als Standarddienste bereit. Durch die Verwendung dieser Services muss die darüber befindliche Sicherungsschicht (manchmal auch Verbindungsschicht) eine einwandfreie Übertragung von Datenblöcken (im Umfang von hunderten von Bits), welche Frames genannt werden, sicherstellen. Da die Bitübertragungs-schicht keine einzelnen Bits interpretiert, muss die Sicherungsschicht den Anfang und das Ende des Frames sowie dessen Einzelteile korrekt definieren.

In der Tat können zahlreiche Unterbrechungen und Fehler auf dem Transferpfad auftauchen, welche dazu führen, dass empfangene Bit-Werte von den gesendeten abweichen. Da die Bitübertragungs-schicht sich jedoch nicht mit einzelnen Bits beschäftigt, werden diese Fehler erst in der Sicherungs-schicht festgestellt. Diese Schicht überprüft die gesamten Frames darauf, ob sie korrekt übertragen wurden (gemäß der Prüfsumme, siehe Kapitel 3). Darüber hinaus stellt sie dem Sender Informationen über die einwandfreie Übertragung von Frames zur Verfügung, wobei sie gleichzeitig – im Falle von beschädigten Frames – deren erneutes Senden erfordert.

3. Netzwerkschicht

Die Sicherungsschicht stellt die Übertragung ganzer Frames sicher, jedoch nur zwischen zwei Netzwerkknoten, zwischen welchen eine direkte Verbindung besteht. Bleibt die Frage, was zu tun ist, wenn man davon ausgeht, dass die Verbindung zwischen einem

Empfänger und einem Sender nicht direkt, sondern über einen oder mehrere zwischengeschaltete Knoten läuft? Hier kommt die Netzwerkschicht ins Spiel, welche ein komfortables Routing der übertragenen Frames, welche als Pakete bezeichnet werden, sicherstellt. Die Netzwerkschicht wählt den korrekten Pfad (oder die korrekte Route) über zwischengeschaltete Knoten aus und stellt darüber hinaus den progressiven Transfer einzelner Pakete auf der Route vom ursprünglichen Sender hin zum finalen Empfänger sicher. Deshalb muss die Netzwerkschicht eine bestimmte Netzwerktopologie beachten (sprich den Weg der direkten Verbindung einzelner Knoten).

4. Transportschicht

Die Netzwerkschicht stellt der darüber liegenden Schicht Dienste bereit, welche den Pakettransfer zwischen zwei willkürlich gewählten Netzwerkknoten sicherstellt. Daraus folgt, dass die Transport- schichtaus der tatsächlichen Netzwerktopologie herausgefiltert wird und die Illusion erzeugt, dass jeder Netzwerkknoten eine direkte Verbindung zu jedem beliebigen anderen Netzwerkknoten hat. Folglich stellt die Transportschicht nur eine Ende-zu-Ende-Kommunikation sicher, sprich die Kommunikation zwischen dem ursprünglichen Sender und dem finalen Empfänger.

Darüber hinaus sorgt die Transportschicht dafür, dass im Zuge der Datenübertragung einzelne Pakete gebildet werden, in welche die übertragenen Daten unterteilt werden. Auf dieselbe Weise werden die Daten nach ihrem Erhalt aus den Paketen entnommen und wieder in ihre ursprüngliche Form gebracht. Dadurch wird die Übertragung eines Inhalts beliebiger Größe sichergestellt, obwohl die einzelnen Pakete eine begrenzte Größe haben.

5. Sitzungsschicht

Diese Schicht stellt Sitzungen zwischen Endteilnehmern her, hält diese aufrecht und unterbricht sie. Während des Sitzungsaufbaus erfordert diese Schicht eine Verbindung zur Transportschicht, über welche eine Kommunikation zwischen den beiden Sitzungsteilnehmern herstellt wird. Diese Schicht ist für ein fallweises Kommunikationsmanagement zuständig (sprich den Entwurf des Übermittlungs-zeitplans beim gleichzeitigen Auftauchen zweier Teilnehmer). Darüber hinaus ist die Schicht für alles zuständig, was mit dem Abbruch der Sitzung und der bestehenden Verbindung zusammenhängt

6. Darstellungsschicht

Die Daten, welche über das Netzwerk übertragen werden, können unter anderem textuell oder numerisch sein oder bloß gewöhnliche Datenstrukturen. Einzelne Knotencomputer können jedoch verschiedene interne Darstellungen dieser Daten verwenden. So nutzen zentrale Computer der Firma IBM einen EBCDIC-Zeichencode, während die meisten anderen mit einem ASCII-Code arbeiten. Ebenso kann ein Computer Integer im Zusatzcode anzeigen, während sie bei anderen vielleicht im direkten Code aufscheinen

usw. Diese Schicht ist für die erforderliche Konvertierung der übertragenen Daten zuständig.

Diese Schicht übernimmt auch potentielle Komprimierungen übertragener Daten oder alternative deren Chiffrierung.

7. Anwendungsschicht

Endnutzer greifen über verschiedene Netzwerkanwendungen auf Computernetzwerke zu: elektronische Postsysteme, Datentransfer, Fernanmeldung usw. Klarerweise braucht man angesichts ihrer enormen Vielfalt nicht einmal an das direkte Einbauen all dieser verschiedenen Anwendungen in die Anwendungsschicht zu denken. Aus diesem Grund umfasst die Anwendungsschicht nur einen Teil dieser Anwendungen, welche übliche, sprich allgemein anwendbare Mechanismen zur Verfügung stellen. Nimmt man die elektronische Post (Mail) als Beispiel, so sieht man, dass ein bestimmter Teil, welcher Netzwerkmeldungen sicherstellt, auch ein integraler Bestandteil der Anwendungsschicht ist. Bei allen Knoten-Computern, welche dasselbe elektronische Mailsystem verwenden, ist dieser Teil identisch. Die Benutzerschnittstelle des elektronischen Mailsystems, sprich eben jener Teil, welcher tatsächlich vom Benutzer verwendet wird und es diesem erlaubt, Nachrichten zu lesen, darauf zu antworten, neue zu verfassen und diese zum Versand aufgibt, wird nicht länger als ein Teil der Anwendungsschicht betrachtet, da sich dieser in jedem bestimmten Knoten deutlich unterscheiden kann, zB im Hinblick auf die Bedienung (durch Zeilenbefehle, verschiedene Menüarten, mit oder ohne Fenster usw.). Ein weiteres typisches Beispiel wäre eine Terminalemulation, welche für eine Fernanmeldung benötigt wird. Alles in allem gibt es eine große Zahl an verschiedenen Terminals, was es fast unmöglich macht, die erforderliche Anpassung zwischen zwei beliebigen Terminalarten zu erreichen. Die Konsequenz daraus ist, dass nur ein „Referenzterminal“ – ein so genanntes virtuelles Terminal – implementiert wird. Jede bestimmte Terminalart besitzt nur eine Anpassung an eben dieses virtuelle Terminal. Nichtsdestotrotz finden sich die Mittel, um mit diesem virtuellen Terminal zu arbeiten, in der Anwendungsschicht (da diese identisch sind), während sich die Mittel zur Anpassung an ein bestimmtes Terminal nicht in der Anwendungsschicht befinden.

6. Drahtlose Kommunikationstechnologien

Kabellose Netzwerke; Grundsätze, Standards, Komponenten, Kommunikations-art, Anwendung und Eigenschaften.

Standards

Die Entwicklung drahtloser Netzwerke ging ziemlich ähnlich vonstatten wie jene von Kabelnetzwerken. Erfolgte sie zu Anfang noch relative spontan, wurde es bald notwendig, Standards zu definieren, welche eine allgemeine Netzwerkkooperation ermöglichten. Die wichtigsten Hersteller drahtloser Technologie schlossen sich zur WECA zusammen (Wireless Ethernet Compatibility Alliance), die Vorgaben betreffend dessen effektives Management machte und auf diesem Weg eine allgemeine Kompatibilität sicherstellte. Erfüllt ein Produkt diese festgelegten Voraussetzungen, erhält es ein WiFi-Zertifikat, welches bescheinigt, dass es mit Produkten anderer Hersteller kompatibel ist. Der kabellose Standard selbst basiert auf dem Ethernet, weshalb sie ähnliche Merkmale haben, sprich eine CSMA/CD Zugriffsmethode und einen ähnlichen Paketaufbau. Es gibt zahlreiche Standards für drahtlose LAN-Netzwerke, dessen Eigenschaften in der Tabelle skizziert werden. Der 801.11g-Standard ist rückwärts-kompatibel mit dem älteren und langsameren 802.11b (sie können miteinander arbeiten, allerdings nur mit geringerer Geschwindigkeit). Der 802.11i-Standard basiert auf dem 802.11g-Standard, wobei der ältere von beiden einen sichereren Authentifizierungs- und Chiffrierungsalgorithmus nutzt.

Komponenten

Drahtlose Komponenten kommunizieren auf zwei nützliche Arten:

- Ad hoc, was sich auf eine direkte Verbindung zwischen mehreren Computern bezieht, sprich zwei bis fünf. Jeder Computer kommuniziert mit einem anderen auf derselben Ebene, sprich sie sind gleichwertig (die Organisation ist einem Peer-to-Peer-Netzwerk ähnlich). Die größten Vorteile sind die schnelle Installation und der geringe Preis (bis auf Client-Netzwerkadapter benötigt man keine andere Hardware). Es ermöglicht das Teilen von Dateien und des Internetzugangs, das Drucken über das Netzwerk und andere Dinge, welche gemeinhin in LANs auftauchen. Auf der anderen Seite ist ein wesentlicher Nachteil, dass alle verbundenen Geräte in einer bestimmten Reichweite sein müssen, sprich jedes Gerät muss das andere sehen. Eine weitere Schwachstelle ist die lächerlich einfach herzustellende Verbindung, welche man nicht als besonders sicher betrachten kann.

- Der Infrastrukturmodus basiert auf einem Access Point (AP). Dieser fungiert als Server, über welchen der gesamte Datenfluss zwischen Netzwerk-Clients stattfindet (die Organisation ist ähnlich zum Client-/Server-Netzwerk). Der größte Vorteil seines Einsatzes ist die Möglichkeit, die Operation zu filter und zu überwachen sowie das Bereitstellen von Netzwerken für verschiedene Clients. Somit wird si-

chergestellt, dass unbewusste Versuche, eine Ad-hoc-Verbindung herzustellen, unterbunden werden. Der gesamte Datenfluss muss zum AP geleitet werden, welcher einen adäquaten Netzwerkschutz gewährleistet. Der Zugangspunkt ist definitiv nicht notwendig, wenn man davon ausgeht, dass eine drahtlose Verbindung nur gelegentlich und nur zwischen ein paar Geräten hergestellt wird. Wird jedoch ein Heimnetzwerk aufgebaut oder in einem Haushalt bzw. einem kleinen Büro geteilt, wird ein Zugangs-punkt in der Regel benötigt (höhere Netzwerksicherheit).

Access Point

Ein AP (Zugangspunkt) ist die Grundlage für ein drahtloses Netzwerk. Er stellt eine Verbindung zwischen drahtlosen Endpunkten und einem Server her, indem er ihn innerhalb eines metallischen LAN-Netzwerks platziert. Deshalb enthält ein AP eine Funkkomponente – Transmitter/Receiver und einen Kabelteil, sprich eine RJ-45-Buchse für einen Twisted-Pair-Anschluss. Zu einem AP gehören Hubs, welche das Signal übertragen. Viele Hersteller bieten den Betrieb des APs mittels Twisted-Pair-Kabel an, über welches der Zugangspunkt mit einem festen Netzwerk verbunden ist. Daher muss der Zugangspunkt (an einer schwer zugänglichen Stelle) nicht die Last zweier Kabelleitungen tragen. Der Zugangspunkt und seine Gegenstücke – Client-Adapter funktionieren, solange es kein Hindernis gibt – zwischen dem Zugangspunkt und den Computern, welche sich im drahtlosen Netzwerk befinden, müssen deutlich sichtbar sein. Eine Folge davon ist, dass man Zugangspunkte vor allem in den obersten Teilen von Räumen finden kann. Bei ihrer Positionierung muss man auf die Ursachen von Funksignalinterferenzen achten, sprich Metallkonstruktionen (auch in Wänden), elektronische Interferenzen (zB funktionieren Mikrowellen in einem Intervall von 2.4 GHz, Mobiltelefone, drahtlose Lautsprecher usw.). Solange eine direkte drahtlose Verbindung benötigt wird, lässt sich eine Wireless Bridge (WB) einsetzen – Zugangspunkte einer Brückenfunktion – welche Pakete zwischen Netzwerken herausfiltert.

Client-Adapter

Dieser Begriff bezeichnet jene Einheit, über welche ein PC mit einem Zugangspunkt verbunden ist. Im Grunde bezieht er sich auf die Netzwerkkarte (mit einer Antenne). Entwickelt wurde er für PCI- und USB-Anschlüsse. Laptops können eine Netzwerkkarte gemäß eines PC-Karten-Standards einsetzen. Allerdings besitzen die meisten Laptops bereits eine integrierte Drahtlosnetzwerkschnittstelle (welche das Aufsetzen und Abgrenzen eines drahtlosen Netzwerks erleichtert).

6.1. Eigenschaften drahtloser Netzwerke

6.1.1. GESCHWINDIGKEIT

Im Gegensatz zu metallischen Netzwerken ist deren Geschwindigkeit erheblich geringer. Die theoretisch maximal erreichbare Geschwindigkeit ist in der Tabelle vermerkt, in welchen auch adäquate Standards bereitgestellt werden. Tatsächlich sind diese Standards schwer zu erreichen, ein Umstand, welcher der schwächeren Erreichbarkeit des Signals zwischen den Funkstationen geschuldet ist, was wiederum dazu führt, das automatisch auf eine geringere Übermittlungsgeschwindigkeit umgestellt wird.

Alles in allem bedeutet das, dass seine längere Strecke (zB mehr als 20 Meter) oder ein Hindernis (zB ein dicker Metallrahmen) die Verbindungsgeschwindigkeit dramatisch reduzieren – um ein Drittel oder um die Hälfte. Solange die Gruppe von Empfängern mit demselben Zugangspunkt verbunden ist, womit sie sich physisch in einem Netzwerksegment befinden, muss die Kapazität von einem Adapter aufgeteilt werden. In der Folge treten viele Kollisionen auf (CSMA/CD Charakter). Eine weitere Variable, welche die realisierbare Geschwindigkeit verringert, ist die sorgfältige Protokollverwaltung in den oberen Schichten, bedingt durch welche sich die tatsächliche Bandbreite für den Transfer reiner Daten merklich reduziert. Gleichwohl beträgt die maximale Bandbreite für die Nutzdatenbelastung unter idealen Bedingungen ungefähr die Hälfte der Nominalwerte, sprich 5 Mbit/s bei einer B-Leitung und 25 Mbit/s bei einer G-Leitung.

6.1.2. SICHERHEIT

SSID (Service Set ID)

Dieser Begriff bezieht sich auf einen Zugangspunkt, welcher für alle Clients sichtbar ist, welche sich innerhalb seiner Reichweite befinden. Aus diesem Grund dient SSID als logischer Indikator für ein bestimmtes drahtloses Netzwerk. Eine SSID kann manuell an der Station angebracht werden. Alternativ kann der Zugangspunkt regelmäßig Informationen über SSID übermitteln oder die SSID-Übermittlung kann in der Situation unterbrochen werden, in welcher der Client selbst nach der SSID fragt.

WEP (Wired Equivalent Privacy)

Dieser Begriff bezeichnet ein altes Sicherheitssystem drahtloser Netzwerke nach dem ursprünglichen IEEE 802.11-Standard. Der Zweck des WEP war es, größtmögliche Sicherheit für verkabelte Computernetzwerke zu gewährleisten (zB Twisted Pair), da das Funksignal leicht selbst über eine längere Distanz hinweg ohne physischen Kontakt mit einem Computernetzwerk aufgefangen werden kann. WEP wurde im August 2001 jedoch geknackt, weshalb es durch das dem IEEE 802.11i-Standard entsprechenden WPA2 ersetzt werden sollte.

WPA (Wi-Fi Protected Access)

Dieser Begriff bezeichnet ein Sicherheitssystem drahtloser Netzwerke. Um das Sicherheits-system WEP im Jahr 2001 zu ersetzen, verbesserte die Wi-Fi Alliance das Sicherheitssystem WPA als Teil des damals im Jahr 2002 vorbereiteten Standards IEEE 802.11i. Überdies hält die Wi-Fi Alliance die Rechte an den Marken Wi-Fi und WPA und zertifiziert ihre Produkte.

WPA2

Es implementiert alle notwendigen Komponenten des IEEE 802.11i. Es nutzt die Blockverschlüsselung des Advanced Encryption Standard (AES), wohingegen die früheren WEP und WPA die aktuelle RC4-Chiffrierung verwenden. Die Architektur von 802.11i enthält die folgenden Komponenten: IEEE 802.1X für die Bestätigung (sprich Extensible Authentication Protokoll – EAP).

Wireless PAN

WPAN verbindet einzelne Geräte in einem relativ kleinen Gebiet miteinander, auf welches eine Person, welche mit dem Netzwerk verbunden ist, leicht zugreifen kann. So können Kopfhörer zB via Bluetooth oder Infrarot mit einem Laptop verbunden sein. Auf diese Weise wird ein kleines privates Drahtlosnetzwerk erzeugt (WPAN). Zig Bee unterstützt auch WPAN-Anwendungen. In der Tat haben sich private Wi-Fi-Netzwerke aufgrund der integrierten Ausstattung einer großen Bandbreite von elektronischen Geräten zu einem Gemeingut für den typischen Konsumenten entwickelt. Darüber hinaus erleichtern Geräte wie “My Wi-Fi” von Intel und “Virtual Wi-Fi” von Windows 7 das Aufsetzen und Konfigurieren eines privaten Drahtlosnetzwerks.

Wireless WWAN

Der Begriff WWAN bezieht sich auf ein drahtloses Netzwerk, welches große Gebiete abdeckt. Diese Netzwerke können zum Verbinden von Geschäftsstellen oder als öffentliches Zugangssystem verwendet werden. Eine drahtlose Verbindung kommt in der Regel durch eine Punkt-zu-Punkt-Mikrowellenleitung zustande, welche einen Parabolspiegels mit einer Frequenz von 2.4 GHz verwendet. Ein typisches System läuft auf den Eingängen zu Basis-stationen, Zugangspunkten und drahtlosen Signalbrücken. Die andere Konfiguration umfasst Netzwerksysteme, in welchen jeder Zugangspunkt das Signal weitergibt.

Wireless MAN

Wireless Metropolitan Networks WMAN bezeichnen drahtlose Netzwerke, welch emit mehreren lokalen Netzwerken verbunden sind. WiMAX ist eine Art drahtloses MAN Netzwerk und wird durch den IEEE 802.16-Standard definiert.

6.1.3. ANWENDUNG

Angewendet werden sie vor allem von Mobiltelefonen, welche ein integraler Bestandteil der täglichen Drahtloskommunikation sind und die Kommunikation erleichtern, die über inter-kontinentale Netzwerksysteme erfolgt, welche auf der ganzen Welt Kommunikations-satelliten einsetzen. Gemeinhin nutzen Privatpersonen und Geschäftsleute drahtlose Netzwerke zum schnellen Versenden und Teilen von Daten, unabhängig davon, ob sie sich in einem kleinen Büro oder irgendwo anders auf der Welt befinden.

7. Virtuelle Netzwerke (VLAN).

VLAN ist die Abkürzung für Virtual Local Area Network. Der Begriff bezeichnet ein funktionales LAN, in welchem ein anderes, ein virtuelles Netzwerk, gebildet wird. Es läuft nur auf einem Hardware-Gerät (physische Verkabelung). Kurz gesagt operiert ein VLAN auf der Grundlage der individuellen Voreinstellung bereits aktiver Netzwerkkomponenten eines neuen virtuellen Modus und Modells.

Ein VLAN arbeitet nach Grundsatz des Taggens von Daten, welche über das Netzwerk übertragen werden. Diese Daten werden gemäß des entsprechenden virtuellen Netzwerks getaggt. Das Hauptziel ist es, mehrere lokale LANs ordnungsgemäß auf der Ebene der zweiten Netzwerkschicht des ISO/OSI-Modells miteinander zu verbinden. Darüber hinaus ist ein großer Vorteil von VLANs deren gute Konfigurierbarkeit.

Unter dem Gesichtspunkt einer qualitativ hochwertigen Netzwerksicherheit lässt sich sagen, dass ein VLAN leicht zwischen Nutzern einzelner Ende-zu-Ende-Stationen und permanent laufenden Anwendungen (z.B. Linux Demon) unterscheiden kann, unabhängig vom aktiven Netzwerkgerät.

Dadurch ist der unautorisierte Zugriff eines unethisch handelnden Hackers auf ein Computernetzwerk völlig ausgeschlossen. Die gängigste Art eines Cyber-Angriffs zielt somit auf die Computernetzwerke selbst ab, wobei einer DDoS-Attacke zu Anfang widerstanden werden kann.

7.1. Wozu braucht man ein VLAN?

Zum ersten Mal veröffentlicht wurde die VLAN-Technologie Mitte der 1990er. Einer von vielen Gründen in diesem Zusammenhang war die Bildung einer Gruppe von bestimmten Nutzern, welche auf einem einfacheren Weg erfolgreich miteinander kommunizieren und auf ihre Daten und Informationen zugreifen können sollten. Ein weiterer Grund war die Stagnierung bei der Entwicklung der Ethernet-Technologie.

Von Relevanz ist auch die Tatsache, dass die VLAN-Technologie problemlos eine Netzwerkkommunikation erkennt, welche vor kurzem ein Netzwerk auf einem geeigneten Netzwerkgerät geteilt hat – Verteiler. In der Tat stellt die Nutzung eines Verteiler-Anschlusses die effektivste und praktikabelste Lösung dar.

Die VLAN-Technologie wurde 1995 entwickelt. Sie basierte jedoch nur auf einer Close-Source-Technologie. Bis vor ein paar Jahren jedoch wurde es hauptsächlich für mittlere und Großunternehmen entwickelt, obwohl bereits ein geeigneter Standard definiert wurde.

Hauptgründe für den Aufbau eines VLANs:

- Einteilen von Netzwerknutzern in Gruppen, Bereiche oder nach bestimmten Diensten anstatt nach einer physikalischen Position und Kommunikationsabschnitten zwischen diesen Gruppen.

- Verringerung von Netzwerkübertragungen, welche vor Jahren zum Problem wurden
- Rechtzeitiges Reduzieren der Collision Domain, welche auftritt, wenn Netzknoten anstelle von Netzwerkweichen verwendet werden.

Die Idee von einer logischen Gruppe von Nutzern wird in der Literatur des Öfteren erwähnt. Ihr zufolge wird bei der Bildung eines VLANs Folgendes beachtet:

- **Organisationelle Struktur:** Die Voraussetzung, dass die Kommunikation innerhalb des Bereichs mit Druckern, Datenservern usw. realisiert wird. Die Voraussetzung, dass es keine Kommunikation zwischen einzelnen Abschnitten gibt. Nur ein paar Dienste (zB Mail) sind üblich.

- **Dienste:** VLAN vernetzt Mitarbeiter, welche dieselben Dienste nutzen (zB Buchhaltung, DB usw.)

Die größen Vorteile von VLAN

Da sich die VLAN-Zugehörigkeit auf verschiedene Arten definieren lässt, werden VLANs typischerweise in Netzwerke mit Port-basierter und regelbasierter Zugehörigkeit unterteilt. Eine etwas differenziertere Unterteilung kennt vier Arten von VLAN-Zugehörigkeit:

- Ports,
- MAC-Adressen von Netzknoten,
- Netzwerkprotokoll oder Netzwerkadressen von Netzknoten,
- Gruppe der IP-Übertragung

Port-basiertes VLAN

Diese historisch gesehen erste virtuelle Netzwerkart legt die Netzwerkzugehörigkeit einzelner Verteiler-Ports (Portgruppen) fest. Die ersten VLAN-Umsetzungen erlaubten keine Erweiterung des virtuellen Netzwerks um mehrere Verteiler. Sie konzentrierten sich auf eine Netzwerkweiche. Auch die darauffolgende Generation erlaubte dies noch nicht. Nachfolgend ist ein solches definiertes Netzwerk abgebildet.

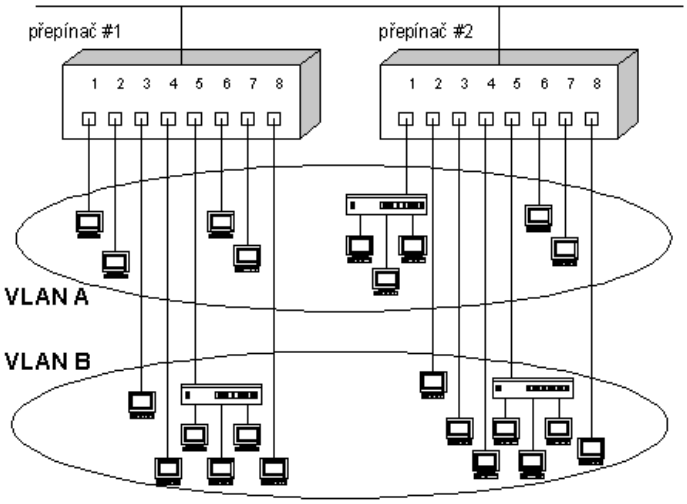


Abb.: Port-basierte VLAN-Zugehörigkeit

Das Gruppieren von Ports ist das Standardprinzip bei der Erstellung eines virtuellen Netzwerks. Obwohl es sehr klar und einfach ist, besteht seine Haupteinschränkung darin, die Zugehörigkeit im Zusammenhang mit jeglicher Verschiebung der Nutzerstation zwischen einzelnen Verteilerports neu definieren zu müssen (sprich solche Veränderungen hätten eine Veränderung der Zugehörigkeit im virtuellen Netzwerk zur Folge).

MAC-Adressen basiertes VLAN

Eine MAC-Adresse ist im Netzwerkadapter-Kreislauf „in Bedrängnis“. Deshalb kann man auf diese Weise definierte virtuelle Netzwerke als nutzerbasierte VLANs betrachten. Geht man davon aus, dass sein Nutzer seine Verbindung ändert (sprich seine Station an einen anderen Verteilerport verlagert), bleibt seine VLAN-Zugehörigkeit dennoch dieselbe.

Diese Methode erfordert eine starre, manuelle Zuweisung der Zugehörigkeit aller Netzwerke einer Station. Ein weiterer großer Nachteil ist die Gefahr einer substanziellen Leistungs-verringerung im Falle eines geteilten Segments, in welchem Stationen verschiedener VLANs mit dem Verteilerport verbunden sind. Das nächste, wenn auch eher kleine, Problem kann auftreten, wenn Nutzer und ihre Laptops ihren Standort ändern und sich über eine permanent angeschlossene Docking-Station mit dem Netzwerk verbinden. Dadurch ändert sich mit dem Standort auch die festgelegte MAC-Adresse (der Netzwerk-Adapter ist im Wesentlichen ein Teil des Docks). Obwohl dies nur ein kleines Problem darstellt, treten bestimmte VLAN-Grenzen zutage, die sich durch eine MAC-Adressen basierte Zugehörigkeit ergeben.

Darüber hinaus sorgt die Möglichkeit der Nutzer, ihre eigene MAC-Adresse direkt im Betriebssystem umzustellen, für weitere Schwierigkeiten, vor allem in Hinblick auf die Sicherheit.

Protokoll- oder adressenbasiertes VLAN

Solche virtuellen Netzwerke basieren auf den Informationen der dritten Netzwerkschicht des OSI-Modells. In Netzwerken mit mehreren Protokollen können Netzknoten nach ihren operierenden Netzwerkprotokollen in einzelne VLANs eingeteilt werden, oder zB in Netzwerke mit TCP/IP-Protokoll entsprechend der Subnetzwerk-Adresse.

Obwohl es hier um Informationen über die Netzwerkschicht geht, ist es wichtig festzuhalten, wir uns nicht mit deren Nutzung für das Routing beschäftigen. Obwohl der Verteiler das Paket überprüfen muss, um die IP-Adresse und damit auch die VLAN-Zugehörigkeit zu spezifizieren, werden keine Routing-Berechnungen durchgeführt. Der Verteiler nutzt keine Routing-Protokolle (wie RIP, OSPF). Folglich muss das VLAN, welches entsprechend den Informationen der dritten Netzwerkschicht definiert wurde, als Netzwerk mit einer flachen Topologie betrachtet werden, das mittels Verteilern oder Brücken zusammengeschaltet ist.

Das gleichmäßige Verteilen in der dritten Netzwerkschicht und der Einsatz von Verteilern mit eingebauter Routingfähigkeit verringern auf den ersten Blick das Problem. Dieses betrifft jedoch verschiedene Funktionen: Einerseits das bloße Ermitteln der VLAN-Zugehörigkeit basierend auf der Netzwerkadresse, andererseits die vollständige Nutzung der Router-fähigkeiten auf Grundlage der Routingprotokolle und Berechnungen. Es ist auch wichtig zu sagen, dass die Kommunikation zwischen einzelnen VLANs den Einsatz von Routern erfordert – egal, ob es sich um klassische Router oder Verteiler mit eingebauten Routing-Funktionen handelt. Gleichzeitig hat das Definieren eines Netzwerkschicht-basierten VLANs eine Reihe von Vorteilen, zB die Nutzermobilität. Genauer gesagt bezieht sich das auf ihre Stationen,

- für welche ein erneutes Konfigurieren der VLAN-Zugehörigkeit nicht notwendig ist,
- welche Gruppen bilden können, die bestimmte Dienste bereitstellen
- oder das (Ab-)Schaffen der Notwendigkeit des Pakettaggings, welches Informationen über die VLAN-Zugehörigkeit enthält, wenn sie miteinander kommunizieren (siehe nachfolgende Absätze).

Gruppenfunkbasiertes VLAN

Der Gruppenfunk in IP-Netzwerken (IP Multicast) funktioniert folgendermaßen: Ein Paket, welches für den Gruppenfunk entwickelt wurde, wird an eine bestimmte Adresse geschickt, die als Proxy für eine ausdrücklich festgelegte Gruppe von Netzwerkknoten (IP-Adressen) funktioniert. Das Paket wird an alle Netzknoten gesendet, welche zu dieser bestimmten Gruppe gehören. Diese Gruppe wird auf dynamischer Grundlage gebildet: Innerhalb dieser Gruppen loggen sich die Netzknoten ständig ein und aus. Folglich können alle Stationen als Mitglieder eines virtuellen LANs betrachtet werden, da die Gruppe eine Domain des Rundstrahlbetriebs bilden. Trotzdem unterscheidet sich das Gruppen-

funkbasierte VLAN in zwei markanten Punkten: Die Gruppe wird nur temporär dynamisch gebildet. Daher ist das Netzwerk sehr flexibel und seine Reichweite wird nicht durch Router begrenzt, sprich sie lässt sich erweitern, zB zu einem ausgedehnten WAN.

7.2. Die größten Vorteile von VLAN

Wie sich Kommunikation in VLAN integrieren lässt

Die VLAN-Integration wird in der Regel auf einem Verteiler eingerichtet (nur in besonderen Fällen nimmt eine getaggte Kommunikation den Übertragungsweg eines anderen Geräts). Verteiler, welche VLAN unterstützen, enthalten zumindest ein VLAN. Dies wird standard-mäßig VLAN NO1 bezeichnet, welches sich nicht löschen oder abschalten lässt. Sofern es nicht anders eingerichtet wird, werden alle Ports (sprich die gesamte Kommunikation) in VLAN 1 integriert. Es gibt vier Hauptmethoden, wie sich die Kommunikation in ein VLAN integrieren lässt. In der Praxis kommt jedoch fast nur die erste Methode zur Anwendung.

1. Port-basierte Methode

Ein Verteiler-Port wird manuell und tief in ein bestimmtes VLAN integriert (konfiguriert). Die gesamte Kommunikation, welche über diesen Port stattfindet, gehört zu dem bestimmten VLAN. Das heißt: So lange, bis dem Port ein weiterer Verteiler hinzugefügt wird, werden sich alle damit verbundenen Geräte im selben VLAN befinden. Damit ist es die schnellste und am häufigsten verwendete Methode. Darüber hinaus gibt es keine besonderen Anforderungen, an die VLAN-Integration. Dessen Definition ist in jedem Verteiler lokal-basiert. Darüber hinaus ist diese Methode klar geordnet und leicht zu verwalten.

2. MAC-Adressen basierte Methode

Die Ports werden entsprechend des Ursprungs der MAC-Adresse in das VLAN integriert. Zu diesem Zweck muss eine detaillierte Tabelle erstellt werden, welche eine Liste der MAC-Adressen der einzelnen Geräte enthält – und zwar zusammen mit dem bestimmten VLAN. Der größte Vorteil ist die höchst dynamische Integration: Wird ein Gerät an einen anderen Port angeschlossen, wird es automatisch in das entsprechende VLAN integriert. Es gibt zwei Arten, auf welche sich diese Methode anwenden lässt: Entweder wird die Integration des Ports in das VLAN entsprechend der MAC-Adresse des ersten Frames eingerichtet, womit die Einstellung gleich bleibt, bis der Port abschaltet. Alternativ wird jeder Frame entsprechend der MAC-Adresse extra in das VLAN integriert. Trotzdem ist diese Methode sehr leistungshungrig. Cisco fand jedoch eine Lösung, die VLAN Membership Policy Server (VMPS) genannt wird und welche einen speziellen Server zur Verwaltung von den Tabellen mit MAC-Adressen braucht. Darüber hinaus integriert diese

Methode Ports in das VLAN. Demzufolge müssen – vorausgesetzt, dass mehrere Geräte (max. 20) damit verbunden sind – alle davon im selben VLAN sein.

3. Protokollbasierte Methode (entspricht den Informationen in der 3. Schicht)

Mit dieser Methode wird die protokollbasierte Integration übertragener Pakete festgelegt, zB die Trennung der IP-Operationen von Apple Talk, die IP-Adressen basierte oder raum-bezogene Integration. In der Praxis kommt sie jedoch nur selten vor. Ein Gerät benötigt eine streng festgelegte IP-Adresse und der Verteiler muss bis zur dritten Schicht operieren (oft operiert er sogar bis zur zweiten), was zu einer merklichen Geschwindigkeitsreduktion führt.

4. Authentifizierungsmethode

Diese Methode verifiziert einen Nutzer oder ein Gerät mittels IEEE 802.1x-Protokoll und nimmt diese gemäß der bereitgestellten Information ins VLAN auf. Primär handelt es sich um eine Sicherheitsmethode, welche den Netzwerkzugriff regelt (NAC). Wird sie jedoch erweitert, dient sie auch als VLAN. Die Methode ist auch dank ihrer Vielseitigkeit effektiv, sprich weder ein physisches Gerät noch der Verbindungsort sind von Relevanz. Ein RADIUS- Server, welcher die Nutzeridentität verifiziert, erlaubt auch das Zuordnen von VLAN-Nutzern. Im Anschluss an die erfolgreiche Authentifizierung werden diese Informationen gesendet. Diese Methode erlaubt auch eine spezielle Einstellung, welche dafür sorgt, dass ein nicht authenfizierter Nutzer in ein spezielles Host-VLAN eingegliedert wird. Cisco-Verteiler können einen Einzel-Host-Port haben, über welchen nur ein Gerät eine Verbindung herstellen kann. Ebenso gibt es einen Multi-Host, welcher es mehreren Geräten erlaubt, sich an einen Port anzuschließen, wobei sich der Port nach der ersten Authentifizierung selbst authentifiziert, wodurch alle Geräte miteinander kommunizieren können.

7.3. Grundsätze der VLAN-Kommunikation

Im Wesentlichen gibt es zwei Arten der VLAN-Zugehörigkeit: Die erste davon dreht sich um die Kommunikation innerhalb eines Verteilers, die zweite bezieht sich auf die Kommunikation zwischen den Verteilern.

Auf einem Verteiler basierendes VLAN

Die VLAN-Kommunikation innerhalb eines Verteilers ist in der Tat sehr einfach. Der Verteiler behält in seinem Arbeitsspeicher Informationen über eine bestimmte Kommunikation (Port), welche zu einem bestimmten VLAN gehört. Daraus folgt, dass innerhalb eines Verteilers nur ein korrektes Routing erlaubt ist. In so einem Fall werden einzelne Ports in ein VLAN integriert – und zwar entweder statisch oder dynamisch, wie es oben bereits erwähnt wurde (Optionen 2, 3, 4). Cisco bezeichnet diese Ports als Zugangsports.

Auf mehreren Verteilern basierendes VLAN

In der Tat ergibt sich eine deutlich kompliziertere Situation, wenn es zu beachten gilt, dass die Information über eine bestimmte VLAN-Integration nicht im Zuge ihres Transfers zu einem anderen Verteiler verloren gehen darf, sprich das gesamte Netzwerk dasselbe VLAN nutzen soll – und zwar unabhängig von einer bestimmten Verteiler-Gerät-Verbindung. Überdies kann ein Gerät an zwei Verteilern angeschlossen sein, dann in dasselbe VLAN integriert werden und die benötigte Information kann trotzdem übertragen werden. Dieses Verfahren ist jedoch sehr ineffektiv.

8. URL, X/HTML, HTTP

URL ist die Abkürzung für Uniform Resource Locator. Dieser wird verwendet, um Dokumente im Internet eindeutig identifizieren zu können. Ein Beispiel für eine URL-Website ist:

http://www.adaptic.cz/znalosti/slovnicek/url/

Dieses zeigt den Aufbau einer URL: CZ ist die Top-Level-Domain, die Second-Level-Domain ist Adaptic und die Third-Level-Domain ist WWW. Diese werden durch Punkte voneinander getrennt. Darüber hinaus enthält die URL einen Pfad zu einer Website eines Verzeichnisses mit der mit Slashes gegliederten Struktur /znalosti/slovníček/url/. Der letzte Teil der URL ist ein Protokoll, welches die Abfrage des Webseiten-Servers ermöglicht. In diesem Fall handelt es sich um das HTTP-Protokoll, welches ganz am Anfang steht (http://).

Des Weiteren kann die URL folgende Informationen enthalten

- einen Webseiten-Titel mitsamt dessen Endung enthalten (zB index.htm),
- die Port-Nummer, welche den erforderlichen Dienst identifiziert (sie steht nach der Top-Level-Domain und ist durch einen Doppelpunkt getrennt, zB 80),
- sofern benötigt auch Benutzernamen und Passwort (erscheinen nach dem Protokoll – user-name:password:@).
- URL-Tabs, welche sich auf eine bestimmte Stelle auf der Webseite beziehen (scheint als #tab am Ende einer URL auf)
- Website-Parameter (tauchen hinter dem Namen der Website auf und sind durch ein Fragezeichen separiert, zB ?logged=true) können auch ein integraler Bestandteil einer URL sein.

8.1. Arten von URLs

Die Art der im Beispiel oben gezeigten URL nennt man eine **absolute URL**. Es gibt jedoch auch eine **relative URL**, welche sich auf eine bestimmte Stelle eines aktuellen Dokuments bezieht (eine Website, welche ihre Links enthält). In so einem Fall können ein paar Teile ausgelassen werden. Das heißt: Vorausgesetzt, eine referenzierte Webseite befindet sich auf demselben Server, kann die Bezeichnung des Servers (Domain) in dieser bestimmten URL weggelassen werden. Auf der anderen Seite ist es nicht notwendig, einen Pfad in die URL zu schreiben, solange sich die Webseite im selben Verzeichnis wie die referenzierte Webseite befindet.

Eine gut gepflegte URL wird **cool URL** genannt (auch auf Tschechisch). Gesetztenfalls, dass nur die Einsetzbarkeit von Interesse ist, spricht man von einer **nutzerfreundlichen URL**. Geht man auf der anderen Seite davon aus, dass nur Suchmaschinen berücksich-

tigt werden, ist von einer **seofreundlichen URL** die Rede.

8.2. Die Bedeutung der URL-Gestaltung

Die Form einer URL wirkt sich stark auf deren Sichtbarkeit im und Verwendungsmöglichkeit für das Web aus. Ist ein Domainname zB kurz und prägnant, können sich die Nutzer diesen leichter merken. Darüber hinaus lässt er sich leichter per Telefon diktieren. Enthält eine URL hingegen ein Schlagwort (oder enthalten Teile der URL ein solches), so wirkt sich dies positiv auf deren Auffindbarkeit für eine Suchmaschine aus. Enthält eine URL jedoch eine große Anzahl an Parametern, wirkt sich dies in beiden Fällen negativ aus: URLs, welche in Großbuchstaben geschrieben werden oder sehr lange sind, werden in E-Mails stark gekürzt und das E-Mail-Programm gibt sie unleserlich wieder.

URL

Eine URL wird entweder als eine absolute oder relative Adresse geschrieben. Während die absolute Adresse sich genau mit der URL deckt, bezieht sich die relative Adresse auf einen bestimmten gekürzten Adresseintrag, welcher auf einer Suchmaschine basiert, die den Eintrag der Adresse der aktuellen Website versteht. Zu beachten gilt, dass eine URL auch die Groß-/Kleinschreibung berücksichtigt.

Absolute URL

Jede absolute Adresse besteht aus einem Protokoll und einem Domainnamen. Darauf folgt meist ein Verzeichnispfad und Dateiname. Manchmal enthält die Adresse auch eine Port-Nummer, eine Tab-Bezeichnung und einen Abfrage-String.

Protokolle

HTTP- oder HTTPS-Protokolle übertragen zumeist HTML-Webseiten. Sie werden in Form von http:// und https:// in die URL geschrieben.

Daraus ergibt sich kein allzu großer Unterschied, solange der Server mit der Suchmaschine kommuniziert. Festzuhalten gilt, dass HTTP unverschlüsselt über das Internet übertragen wird, wohingegen HTTPS ein verschlüsseltes Protokoll ist.

In der Tat sollten die beiden Protokolle nicht verwechselt werden, da beide mitunter auf einem bestimmten Server nicht funktionieren könnten. Das heißt: Wann immer absolute Adressen verwendet werden, muss eruiert werden, welche Protokolle in einem bestimmten Web funktionieren.

Domains

Jeder Internetserver hat einen Domainnamen. Dieser besteht aus drei Teilen, welche durch Punkte voneinander getrennt werden:

- Bezeichnung des virtuellen Servers, zumeist www (Third-Level-Domain)
- Bezeichnung der Second-Level-Domain (Registrierung erforderlich)
- Top-Level-Domain, zumeist Länder- oder Firmenkürzel wie CZ, SK oder COM

Port

Auf eine generische Domain folgt selten eine per Doppelpunkt abgegrenzte Port-Nummer.

Verzeichnispfad

Die Zieldatei befindet sich entweder in einem Verzeichnis oder direkt im Hauptverzeichnis des Servers. Verzeichnisse sollten in der URL an die generische Domain angehängt werden. Ein Schrägstrich (kein Backslash) wird vor der Bezeichnung eingefügt. Verzeichnisse mit mehreren Ebenen schreibt man hintereinander und trennt sie durch Slashes voneinander.

Dateien

Ein Dateiname wird nach dem Verzeichnispfad geschrieben (wenn er existiert). Vor dem Dateinamen wird ein Slash eingefügt.

Registerkarte (Tab)

Ein direkter Link stellt eine Verbindung zu einem Tab in einem referenzieren Dokument her. Hierzu schreibt man nach dem Dateinamen eine Raute # sowie den Tab-Namen in die URL.

Anfrage

Eingabedaten für ein bestimmtes Script können auch Teil einer URL sein. Diese werden nach dem Fragezeichen ans Ende der URL geschrieben. Syntax:

```
Name=Wert&Name2=Wert2
```

Beispiel für eine typische absolute URL:

http://www.jakpsatweb.cz/html/url.htm#priklad

Adressteil	Beispiel	Andere mögliche Werte

Protokoll	http://	ftp:// , mailto: etc.
Third-Level-Domain (Server)	www.	www. , xyz.
Second-Level-Domain	jakpsatweb.	seznam. , mujweb. usw.
Top-Level-Domain	cz	com, sk, gov usw.
Port	nichts	:80 , :Zahl
Pfad (Verzeichnisse)	/html/	/, /xyz/Verzeichnis/
Dateiname	url.htm	index.html xyz.html
Registerkarte	#Beispiel	#Registerkartenbezeichnung
Anfragen	nichts	?Variable=Wert

Relatives Adressieren

Oft ist es so, dass das Eintippen einer vollständigen absoluten Adresse ein unnötiger und langwieriger Prozess ist. Man kann sich diesen Aufwand sparen, indem man stattdessen relative Adressen verwendet.

Die Idee der relative Adressen basiert auf bestehenden Dateien, welche miteinander verbunden und auf demselben Server gespeichert sind. Jede Datei, welche eine andere URL-Datei benötigt, hat eine absolute URL. Aus diesem Grund muss nur ein Dateipfad, Slash und Dateiname an die Adresse angehängt werden, woraus sich eine relative URL ergibt.

Relative URL = Pfad/Datei_Name

Verzeichnisse werden durch Schrägstriche voneinander getrennt. Ist es der Fall, dass sich eine Zieldatei höher in der Verzeichnishierarchie befindet (wodurch ein bestimmter „Sprung nach oben“ erforderlich wird), so müssen zwei Punkte als Ersatz für die Bezeichnung des übergeordneten Verzeichnisses eingefügt werden.

Ein Beispiel: Fügt man ein Bild mit dem Logo “Jak psát web” (Wie man ein Web schreibt) auf einer Website ein, so macht man das mithilfe einer relativen Adresse. Dies sieht wie folgt aus: <img src="../images/jakpw.png">

8.3. XHTML

XHTML bezeichnet eine moderne Mark-Up-Sprache, welche die bereits veraltete Sprache

HTML ersetzt hat. XHTML-Dokumente können auch unter Anwendung der Sprache XML umgesetzt werden, wovon es sich in mehrerlei Hinsicht unterscheidet. In diesem Kontext sind beispielsweise die Anforderung an die Code-Deklaration, strengere Regeln betreffend Einträge (müssen geschlossen sein) und Attribute (Kleinbuchstaben in Anführungszeichen) zu nennen.

Heutzutage existieren zwei Versionen von XHTML. Die erste davon ist XHTML 1.0, welche sich wiederum in drei Varianten aufteilen lässt. Dies sind: Frameset (für Websites, die Frames verwenden), Transitional (erleichtert die Umwandlung in XHTML) und Strict (die genaueste Variante). Die zweite Version ist XHMTL 1.1. Im Wesentlichen unterscheidet sich diese kaum von der Strict-Variante von XHTML 1.0. Sie ist nur in mehrere Module unterteilt.

Obwohl die Meinungen darüber auseinandergehen, stellt XHTML im Hinblick auf die Umsetzung von WWW-Websites das praktikabelste Werkzeug dar. Anzumerken gilt, dass XHTML 1.1 nicht mit älteren Browsern kompatibel ist. Auf der anderen Seite geben die zwei verbleibenden Varianten von XHTML 1.0 zu wenige Einschränkungen vor, wodurch sie, wie HTML, hohe Anforderungen an die Disziplin der Programmierer stellen.

XHTML unterscheidet sich vom neueren HTML-Standard. Im Großen und Ganzen hat sich HTML lange Zeit nicht weiterentwickelt. Erst in der HTML-Version 4.01 wurde ein erster Annäherungsversuch an XHMTL unternommen.

Das “X” am Beginn von XHTML steht für eXtensible, sprich erweiterbar. In der Praxis jedoch nimmt man es jedoch eher als “einengend” und “beschneidend” wahr.

Insgesamt wird XHMTL von aktuellen Browsern ebenso unterstützt wie HTML (was 2004 geschrieben wurde, ist auch 2012 gültig). Obwohl man zunächst davon ausging, dass XHTML in Zukunft besser unterstützt werden würde als HTML, so ist in Anbetracht der Erfahrung, welche man mittlerweile betreffend die Browser-Entwicklung sammeln konnte, nicht anzunehmen, dass dies tatsächlich der Fall sein wird.

Unterschiede zwischen XHTML und HTML

Anders als in HTML müssen bei XHTML alle Tags – sogar jene, welche keinen End-Tag haben (zB <meta>, <link>,
, <hr> or <img>) – geschlossen werden. Der Eintrag kann mehrere Formen haben: entweder klassisch (valide) <img></img> oder abgekürzt <img/> oder geringfügig angepasst <img />. Es wird dennoch davon abgeraten, die erste Methode zu verwenden, wenn ein XHTML-Dokument des Typs text/html verschickt wird. Die zweite Schreibweise, jene ohne Lücke, sollte man nicht für ältere Browsern verwenden, da sie das letzte Attribut auslassen könnten, sofern ein solches eingegeben wurde.

Darüber hinaus muss man in XHTML – anders als bei HTML – alle Tags und deren Attribute in Kleinbuchstaben schreiben. Der Grund dafür ist die Unterscheidung zwischen Groß- und Kleinschreibung in DTD und X(HT)ML, mit welchen diese deklariert und referenziert werden. Mit anderen Worten: Die Schriftgröße wird beachtet. Unter der Voraussetzung, dass man ein eigenes DTD deklariert, ist der Einsatz von Großbuchstaben kein Problem.

Alle Attributwerte müssen in Anführungszeichen gesetzt werden.

Darüber hinaus muss ein Dokument mit einer XML-Deklaration beginnen, obwohl deren Einsatz nicht verpflichtend ist, wenn das Dokument in UTF-8 verschlüsselt ist oder die Verschlüsselung von einem höheren Protokoll übernommen wird (zB HTTPS). [14]

Sofern Frames benötigt werden, kann man ein Dokument als XHTML 1.0 Frameset, bei einzelnen Webseiten sogar XHTML Transitional, deklarieren.

Ein XHTML-Dokument sollte auf eine andere MIME-Art versendet werden als ein gewöhnliches HTML-Dokument. [15]

8.4. HTTP

HTTP bezieht sich afu ein Internetprotokoll, welches ursprünglich für den Austausch von Hypertext-Dokumenten zwischen einem Server und einem Browser (sprich WWW-Dienst) konzipiert wurde. In der Tat ist die aktuelle HTTP-Version in der Lage, Dateien jeglicher Art zu übermitteln und sie wird auch für mehrere Zwecke genutzt (zB die Fernbedienung von Anwendungen). Es gibt auch eine sichere Version von HTTPS für HTTP.

HTTP basiert auf dem Prinzip Anfrage Antwort. Individuelle Anfragen sind aus der Sicht des Servers nicht erkennbar. Deshalb wird HTTP auch als zustandsloses Protokoll bezeichnet. Tatsächlich war dies ein großer Vorteil in Zeiten von Internetauftritten. Programmiert man jedoch komplexere Web-Anwendungen, treten vertrackte Probleme auf. So lässt es HTTP zB nicht zu, dass man den Inhalt eines Warenkorbs in einem Online-Shop speichert. Daraus folgt, dass man effektivere Methoden benötigt, um das Problem

in den Griff zu bekommen, zB Cookies.

HTTP-Protokoll

Ein HTTP-Protokoll erschließt einen Kommunikationsweg des Browsers zum Server, während eine Website heruntergeladen wird.

Effektiver Einsatz von HTTP

Um eine Website zu erzeugen, ist es nicht notwendig, Kenntnis über das HTTP-Protokoll zu haben. Sobald es jedoch um komplexere Angelegenheiten geht oder man sich mit einer Suchmaschinenoptimierung auf fortgeschrittener Ebene beschäftigen muss, benötigt man ein tieferes Verständnis von den Aktivitäten, welche zwischen dem Server und dem Client stattfinden.

HTTP-Header stellen zB Informationen über Weiterleitungen, Caching, Cookies, Referrer und Komprimierungen zur Verfügung.

Sobald Scripts oder komplexere Web-Programme geschrieben werden, ist es von Vorteil zu wissen, wohin und wie man eine bestimmte HTTP-Antwort sendet.
Die Abkürzung HTTP steht für Hyper Text Transfer Protocol (Hypertext ist ein Text mit Links).

Funktionsweise des HTTP-Protokolls

Ein Client kommuniziert mit dem Server, sprich der Client will etwasund der Server stellt es zur Verfügung.

- Ein Client wird in der Regel durch einen Internet-Browser (Explorer, Mozilla, Opera) repräsentiert. Ein Suchroboter oder ein anderes Programm können das jedoch auch.
- Ein HTTP-Server enthält ein Programm, welches in einem Server-Bereich auf einem Computer läuft (obwohl ein Computer auch “Server” genannt wird, ist damit nicht der “HTTP-Server” gemeint). Der häufigste HTTP-Server ist das Programm Apache.

Das HTTP-Protokoll ist somit eine Art von Sprache, mit deren Hilfe die zwei Programme miteinander kommunizieren. Dies geschieht über ein Netzwerk, meistens das Internet.

Ein Client benötigt in der Regel eine bestimmte Website. Er verbindet sich mit einem Server und fragt den Server nach URL-Webseiten. So eine Anfrage wird im HTTP-Protokoll formuliert (die Verbindung selbst wird über das TCP-Protokoll hergestellt). Daraufhin stellt der Server die Anfrage zu und sendet eine Antwort zurück, welche ebenfalls im HTTP-Protokoll festgeschrieben ist. So sendet er zB HTTP-Header an einen Client, gefolgt von einem Webseiten-Text in HTML. Der Client erhält die Antwort, liest die Header und zeigt die Website an.

Beispiel einer HTTP-Kommunikation

Ein Leser möchte eine Website anschauen. Ihre URL lautet wie folgt:

https://www.jakpsatweb.cz/server/http-protokol.html.

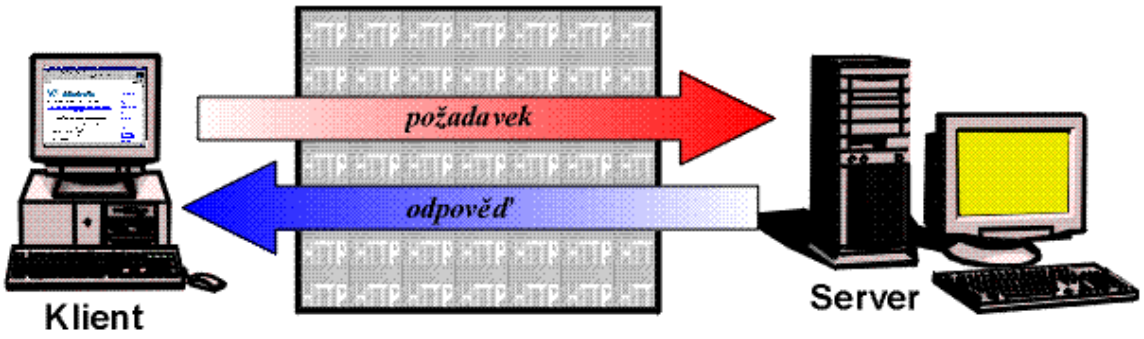
1. Der Leser tippt die URL in den Browser ein.
2. Der Browser (Client) evaluiert die Domain und findet via DNS heraus, nach welcher IP-Adresse gefragt werden soll.
3. Der Client stellt via TCP-Protokoll eine Verbindung zum unter der ermittelten IP-Adresse befindlichen Server her. Hier kommt HTTP ins Spiel.
4. Der Browser sendet folgenden HTTP-Befehl an den Server:

GET /server/http-protokol.html HTTP/1.1
HOST: www.jakpsatweb.cz

HTTP-Protokolle

Der Dienst des World Wide Web basiert im Wesentlichen auf drei Technologien: HTML, URL und http. HTML bezeichnet eine Mark-Up-Sprache, die für die standardmäßige Beschreibung des Inhalts und der Struktur einer Website verwendet wird. URL steht im Zusammenhang mit bestimmten Adressen, welche im Web verwendet werden: Jede Website hat ihre eigene, eindeutige Adresse in Form einer URL. HTTP, das Hypertext Transfer Protocol, ist jenes Protokoll, welches für die Kommunikation zwischen Browsern und Web-Servern verwendet wird. Mit seiner Hilfe werden die vom Benutzer gewünschten URL-Webseiten (über einen Browser) vom Server übertragen. Der Server wiederum schickt per HTTP eine Website an den Benutzer zurück, welche in HTML geschrieben wurde.
Um die Prinzipien von CGI-Skripten zur Gänze zur verstehen, ist zumindest ein rudimentäres Wissen über HTTP erforderlich. Aus diesem Grund werden die Hauptmerkmale von HTTP nachfolgend im Detail angeschaut.

Das HTTP-Protokoll ergibt sich aus der Client/Server-Architektur. Der Client, in diesem Fall ein Browser, verbindet sich mit dem Server und schickt eine Anfrage. Im Gegenzug erhält er eine Antwort vom Server. Das standardmäßige Format von Anfrage und Antwort ist im HTTP-Protokoll festgelegt. Verkompliziert wird das ganze dadurch, dass es drei Protokoll-Versionen gibt: 0.9, 1.0 and 1.1. Daraus folgt, dass sich das Format von Anfrage und Antwort in einzelnen Versionen grundlegend unterscheidet.



Anfrage **Antwort**

Abb.: Der Kommunikationsablauf zwischen Client und Server

Haupteigenschaften von HTTP

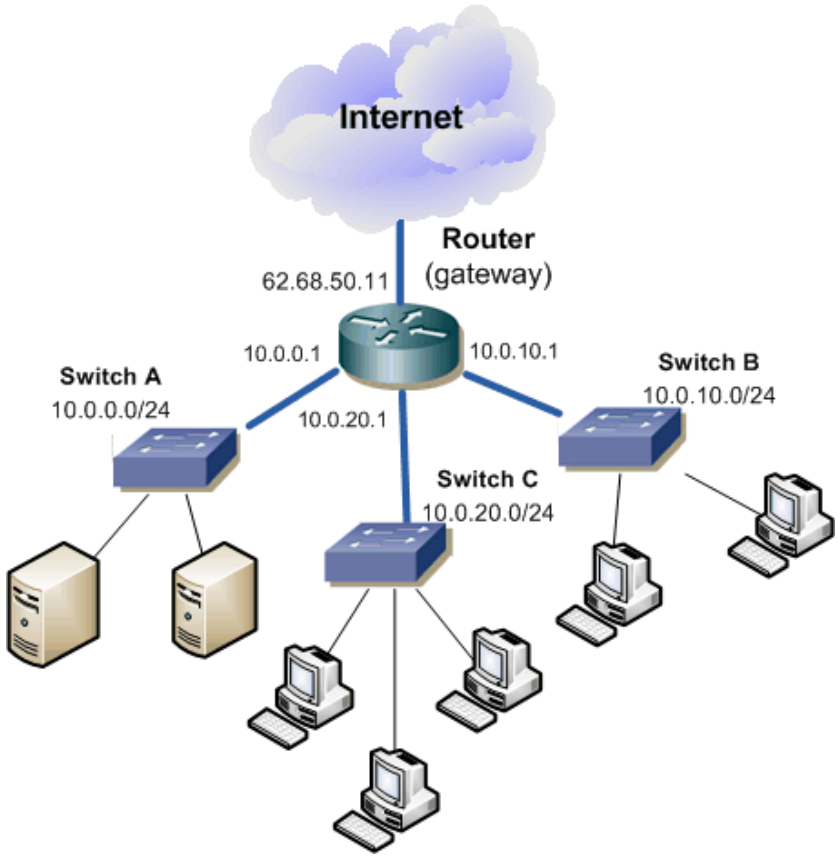
Um dieses Kapitel zur Gänze zu verstehen, ist ein umfassendes Wissen über die Hauptmerkmale von HTTP erforderlich, sprich über die tatsächliche Funktionsweise des Protokolls. HTTP bezieht sich auf die Anwendungsebene verteilter Hypermedia-Informationssysteme. Das heißt, dass dieses Internetprotokoll in der Regel nicht nur für die Datenübertragung zwischen Client und Server verwendet wird, sondern noch eine Reihe weiterer Aufgaben übernimmt. Da HTTP zustandslos ist, unterscheidet es nicht zwischen Clients, von welchen es Anfragen erhält. Sendet beispielsweise ein Client zwei Anfragen gleichzeitig, wird der Server nicht erkennen, dass es sich um denselben Client handelt.

HTTP existiert in drei Versionen: 0.9, 1.0 und 1.1. Die erste davon, welche HTTP/0.9 bezeichnet wird, ist ein einfaches Protokoll, welches in der Lage ist, Daten im Internet eingeschränkt zu übertragen. HTTP/1.0 hingegen erlaubt dem Protokoll, Informationen im MIME-Format zu übertragen, wodurch Daten auch Meta-Informationen enthalten können. Gesamt gesehen wurde die größte Verbesserung jedoch dadurch erreicht, dass alle Verbindungen dauerhaft gemacht wurden. Dies geschah in HTTP/1.1, der bislang aktuellsten Version. Dauerhaft heißt, dass eine Verbindung nicht geschlossen wird, bis ein Teil des Client-Server-Paars einen Beendigungs-Header verschickt. Zuvor beendete HTTP eine Verbindung automatisch nach jeder einzelnen Server-Antwort. Durch diese große Verbesserung hat sich die Übertragungsgeschwindigkeit massiv erhöht, da der Server nicht für jedes Bild, Applet und Frame eine neue Verbindung herstellen muss.

9. Routingprotokolle

Routing bezeichnet eine Methode, welche sich auf das Verknüpfen einzelner Netzwerke fokussiert (Subnetze). Das ursprüngliche Gerät, welches für das Routing konzipiert wurde, war der Router. Heutzutage werden hierzu jedoch hauptsächlich L3-Verteiler, Firewalls oder einfach Server/Computer eingesetzt. Ein Router leitet die Kommunikation von einem Netzwerk an ein anderes weiter.

Die nachfolgende Abbildung illustriert ein einfaches Netzwerk-Beispiel mit den Subnetzen A, B und C. Diese Subnetze sind durch einen Router miteinander und dadurch mit dem Internet verbunden. Daraus lässt sich Folgendes ableiten: Möchte beispielsweise Subnetz B mit dem Subnetz-Server A kommunizieren, sendet dieses Daten an den Router, wodurch deren Auslieferung an Subnetz A sichergestellt wird. Möchte die Station über das Internet kommunizieren, sendet der Router die Daten an eine andere Schnittstelle.



Die Einteilung von Netzwerken in Subnetze basiert auf einer bestimmten Hierarchie, wobei alle Verknüpfungen einen Router haben müssen. Die Kommunikation erfolgt nach oben hin, sprich in die nächstgelegene Schicht, welche bestimmte Subnetze miteinander verbindet, von wo aus sie wieder nach unten verlegt wird. Die Pfadlänge wird in Form von **Hops (Sprüngen)** berechnet, womit einzelne Transite von Gerät zu Gerät ge-

meint sind, sprich die Anzahl von Routern, welche sich auf dem Weg befinden, ergänzt um die Zahl 1. Eine direkte Verbindung zwischen zwei Computern entspricht einem Hop. Der Begriff **„nächster Hop“** wird auch häufig verwendet und bezieht sich auf die Adresse des nächsten auf dem Weg gelegenen Routers.

Die nächste Abbildung illustriert die oben erklärte Situation. Es gibt einen kleinen Abschnitt (nur schematischer Natur) eines größeren Netzwerks oder des Internets. Kleinere Router befinden sich in den Blättern des Baums und sind mit Verteilern und Computern verbunden. Diese Router sind mit anderen (größeren) Routern auf unterschiedlichen Ebenen verbunden. Es ist jedoch so, dass größere Router für gewöhnlich nur die Aufgabe haben, das Netzwerk vor einem Ausfall zu schützen oder die Last zu verteilen.

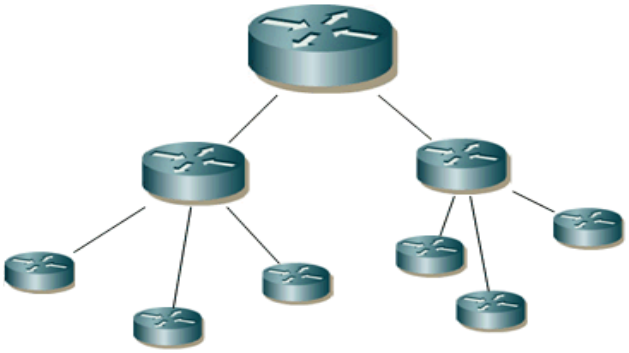


Abb.: Router-Hierarchie im Netzwerk?

9.1. Routing: Technische Begriffe

Router
Ein Gerät, welches das Routing durchführt.

Routing
Der Prozess der Datenweiterleitung zwischen Netzwerken.

Route
Ein angewendeter Pfad, welcher in einer Routing-Tabelle festgeschrieben ist.

Routing-Tabelle
Enthält Aufstellungen über einzelne Routen.

Routing-Protokoll
Verwaltet die Ausrichtung eines gerouteten Protokolls: Es legt die beste Route hin zum Ziel fest und sendet Routing-Informationen an andere Router.
Geroutetes Protokoll

IP, IPX oder Apple Talk. Auch ein wichtiger und häufig verwendeter Begriff.

Router on Stick

Dieser Begriff bezeichnet einen Router, welcher über einen Hauptanschluss mit einem Verteiler verbunden ist. Das bedeutet, es sind nur ein Router und eine Leitung verfügbar, was zu einer beträchtlichen Belastung dieser beiden führt, wodurch letztlich Probleme (Ausfälle) hervorgerufen werden.

Einteilung von Routing-Protokollen

Die Routing-Tabelle enthält einige Aufzeichnungen von Routen, welche unterschiedlichen Ursprungs sein können. Daraus folgt, dass Pakete auf einer der folgenden Routing-Arten geroutet sind:

- Statisches Routing: manuell eingetragene Routen (Aufzeichnungen in der Routing-Tabelle), sicher und von hoher Qualität. Veränderungen in der Netzwerk-Topologie spiegeln sich darin jedoch nicht wieder.

- Dynamisches Routing: Das Netzwerk passt sich automatisch an Veränderungen in der Topologie und beim Transport an. Routen werden automatisch von einem Routing-Protokoll berechnet.

- Standardmäßiges Routing: Solange keine andere Art festgelegt wurde, kommt dieses zum Einsatz.

Dynamische Routing-Protokolle lassen sich wiederum in zwei Arten unterteilen:

- Distanzvektoralgorithmus: Router behalten die Routing-Tabelle bei, welche Informationen über die Distanz (Vektor) eines bestimmten Netzwerks enthält. Darüber hinaus senden sie die Routing-Tabelle an ihre Nachbarn, welche ihrerseits eine eigene Routing-Tabelle erstellen und diese weiterleiten. Um die bestmögliche Route zu berechnen, werden eine (Anzahl der Hops beim RIP) oder mehrere Mess-größen (Leitungsdurchsatz und IGRP-Verzögerung) verwendet. Zusätzlich wird ein aufgewerteter Typ des Distanzvektorprotokolls in Form eines Pfadvektorprotokolls dargestellt.

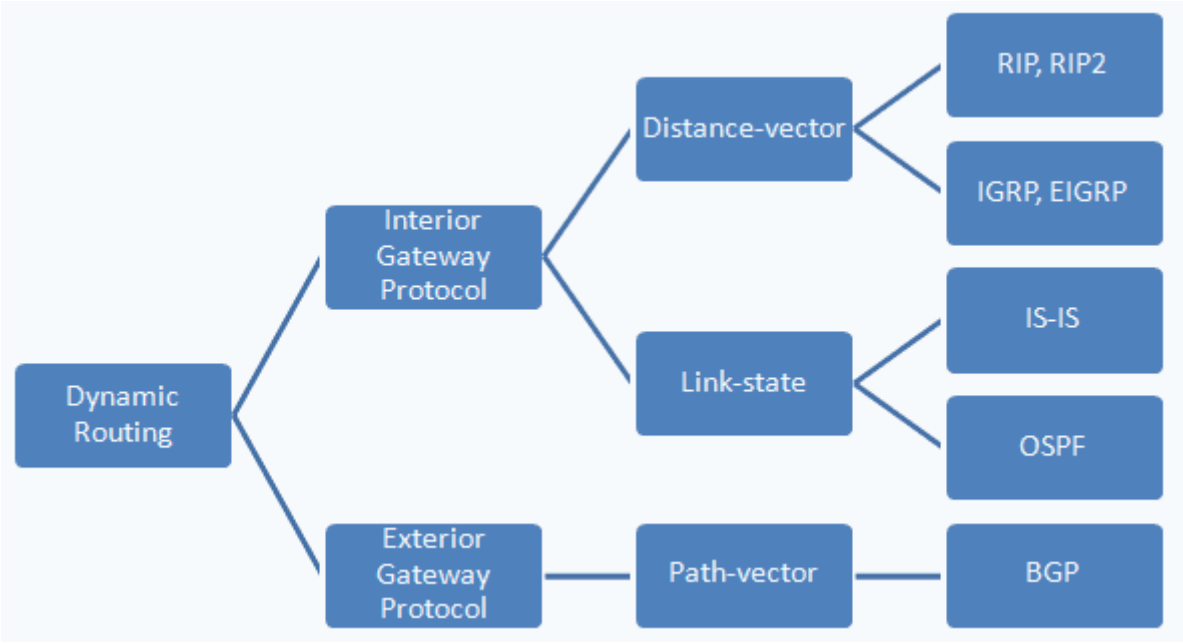
- Link-State Routing-Protokoll: Router behalten eine umfassende Netzwerktopologie-Datenbank bei (erstellt mittels LSA), verändern Link-State-Advertisements (LSA). LSAs werden durch ein bestimmtes Ereignis im Netzwerk ausgelöst. Dieses Protokoll verschickt auch Hello-Pakete, welche Informationen über das Protokoll enthalten und reagiert schnell auf Veränderungen in der Topologie. Gleichwohl dehnt es sich auf einen größeren Bereich aus und verwendet mehr Ressourcen des Routers. Darüber hinaus ist seine Messgröße komplex-basiert und die bestmögliche Route wird vom Dijkster-Algorithmus Shortest Path First (SPF) berechnet.

Anmerkung: Es gibt noch eine weitere Art von dynamischem Routing-Protokoll, welche

zwar auf dem Distanzvektorprotokoll basiert, jedoch auch einige Merkmale des Link-State-Protokolls besitzt. Hierbei handelt es sich um ein hybrides Routing-Protokoll oder ein fortgeschrittenes Distanzvektorprotokoll. Dessen einziger Vertreter ist das EIGRP.

Darüber hinaus lassen sich dynamische Protokolle dahingehend unterteilen, ob sie für einen Einsatz in einem lokalen Netzwerk konzipiert sind (genauer gesagt in einem autonomen System [AS], welches aus mehreren LANs bestehen kann) oder ob sie netzwerkübergreifend operieren (mehrere AS miteinander verbinden).

- Interior Gateway Protocol (IGP): Routing innerhalb Autonomer Systeme (AS)
- Exterior Gateway Protocol (EGP): AS-übergreifendes Routing



10. Sicherheit und Verschlüsselung

10.1. Netzwerksicherheit

Gefahren:

Lauschangriff, Veränderung übertragener Daten, unautorisierter Zugriff auf ein lokales Netzwerk.

Adäquater Schutz:

- Schutz von Daten vor unautorisiertem Erbeuten, Austauschen und Löschen
- Rechenkapazität einzelner Netzwerkknoten
- Zurückfahren der Funktionalität oder Reduktion von Störungen des Dienstverkehrs

Passive Angriffe

- "Lauschangriff", um nicht-öffentliche Informationen zu bekommen, welche missbraucht werden könnten.
- Verkehrsüberwachung: Analyse der Kontakte, welche über ein Netzwerk erfolgten

Aktive Angriffe

- Datenveränderung
- Einschleusen gefälschter Daten
- Vor aktiven Angriffen kann man sich zwar nicht zu 100% schützen. Im Gegensatz zu passiven Angriffen lassen sie sich jedoch einfacher aufdecken.

Ziele von Sicherheitsdiensten

- Sicherstellung der Geheimhaltung von Daten: Durch das Verschlüsseln des gesamten Kommunikationskanals oder ausgewählter sensibler Daten.
- Sicherstellen, dass sich alle Netzwerknutzer authentifizieren müssen (Aufspüren eines getarnten Eindringlings)
- Sicherstellung der Datenintegrität
- Gewährleisten, dass Mitteilungen verweigert werden können: So wird es Nutzern unmöglich gemacht, das Senden und den Empfang einer Nachricht zu leugnen.
- Vergabe von Zugangsrechten, um den Zugriff auf Computer, Daten und Anwendungen zu kontrollieren. Das Identifizieren und Authentifizieren von Anwendern vor deren Zugriff sind ebenfalls ein integraler Bestandteil dieser Vorkehrung.
- Gewährleisten der Verfügbarkeit von Netzwerkdiensten. Angriffe auf die Verfügbarkeit von Diensten können durch Authentifizierung und Verschlüsselung verhindert werden.

10.2. Firewall

- Bezeichnet eine Reihe von Maßnahmen (implementiert durch HW und SW), welche das Netzwerk vor einem unautorisierten Zugriff von außen schützen und gleichzeitig vor einem Informationsleck bewahren.

- Erlaubt zB die Regulierung des Nutzerzugriffs von externen und internen Netzwerken aus, Die Vergabe von Zugriffsrechten, das Herausfiltern gefährlicher Dienste, das

Kanalisieren Der Sicherheit auf einen Kommunikationsknoten, das Blockieren eines feindlichen Mappings des internen Netzwerks sowie das Überprüfen legaler und illegaler Operationen.

- Schützt beim Einstieg ins und Ausstieg aus dem Netzwerk.
- Fungiert als Filter, sprich entscheidet, was und bis zu welchem Punkt etwas erlaubt ist.

10.3. Verschlüsselung

1. Symmetrische Verschlüsselung

Zur Ver- und Entschlüsselung benötigt man einen einzigen Schlüssel. Die symmetrische Verschlüsselung, manchmal auch „gewöhnliche“ Verschlüsselung genannt, bezeichnet einen kryptografischen Algorithmus, welcher einen einzelnen Schlüssel für die Ver- und Entschlüsselung verwendet. Dadurch unterscheidet er sich von asymmetrischen Algorithmen, welche ein Schlüsselpaar verwenden, das aus einem geheimen und einem öffentlichen Schlüssel besteht.

Der große Vorteil symmetrischer Verschlüsselung ist ihre geringe Berechnungskomplexität. Asymmetrische Algorithmen können bis zu einige hundert Mal langsamer sein. Andererseits besteht der größte Nachteil darin, den geheimen Schlüssel teilen zu müssen. Aus diesem Grund müssen sich der Absender und der Empfänger einer geheimen Nachricht auf einen bestimmten geheimen Schlüssel einigen.

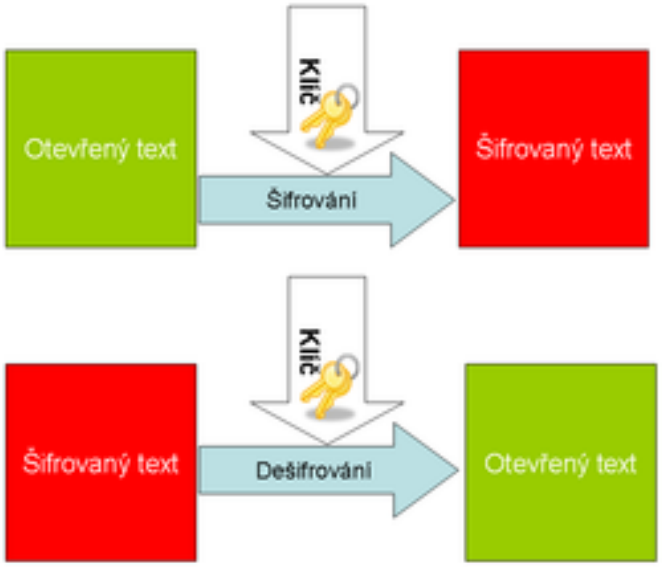


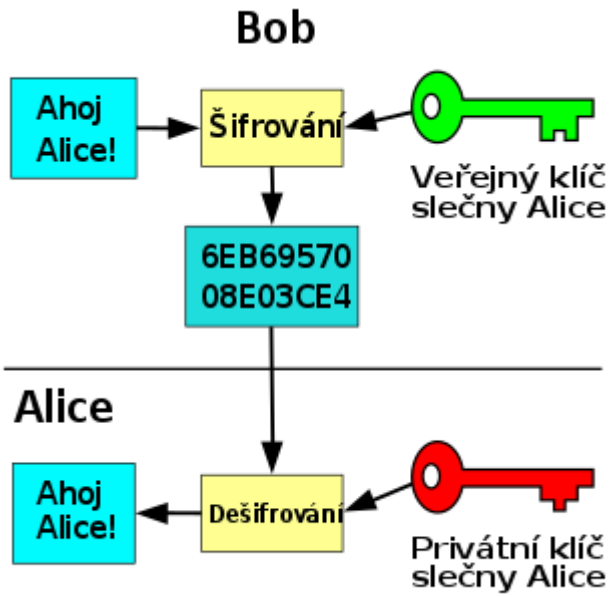
Abb.: Symmetrische Verschlüsselung

Klartext → Verschlüsselung→ verschlüsselter Text

Verschlüsselung → entschlüsseln → Klartext

2. Asymmetrische Verschlüsselung

Beispiele für asymmetrische Verschlüsselung
Asymmetrische Verschlüsselung (Public Key Encryption) bezieht sich auf eine Gruppe von kryptografischen Methoden, die unterschiedliche Schlüssel für das Ver- und Entschlüsseln verwenden. Alles in allem ist dies der Hauptunterschied zur symmetrischen Verschlüsselung, welche nur einen einzigen Schlüssel zum Ver- und Entschlüsseln verwendet.
Die asymmetrische Verschlüsselung nicht nur dazu genutzt, eine geheime Kommunikaion zu gewährleisten, sondern auch für die elektronische Signatur, sprich zur Kenntlichmachung des Verfassers von Daten.



Bob
Hallo, Alice → Verschlüsselung ← öffentlicher Key von Alice

Alice
Hallo, Alice ← Entschlüsselung ← privater Schlüssel von Alice

Abb.: Asymmetrische Verschlüsselung

Der Verschlüsselungsschlüssel bei der asymmetrischen Verschlüsselung besteht aus zwei Teilen: Der erste Teil wird für die Verschlüsselung von Nachrichten verwendet (der Empfänger der Nachricht muss diesen nicht kennen). Der zweite Teil wird zum Entschlüsseln verwendet (der Absender der verschlüsselten Nachricht kennt diesen in der Regel nicht). Klarerweise muss der Absender in diesem Fall keinerlei Geheimnisse mit dem die Botschaft entschlüsselnden Empfänger teilen, weshalb es nicht nötig ist, Schlüssel auszutauschen. Das ist der Hauptvorteil der asymmetrischen Verschlüsselung.

In ihrer gängigsten Variante verwendet die asymmetrische Verschlüsselung einen privaten und einen öffentlichen Schlüssel. Der Schlüssel zur Verschlüsselung ist öffentlich und sein Besitzer stellt diesen zur Verfügung, damit jeder bestimmte Nachrichten verschlüsseln kann. Der Schlüssel zur Entschlüsselung hingegen ist privat, sprich sein Besitzer hält ihn geheim und verwendet ihn, um die Nachrichten zu entschlüsseln (es gibt noch andere Methoden der asymmetrischen Verschlüsselung, welche es erfordern, den Schlüssel geheim zu halten).

Klarerweise müssen der Verschlüsselungs-Schlüssel “e” und der Entschlüsselungs-Schlüssel “d” mathematisch miteinander verflochten sein. Im Hinblick auf die Effektivität der Verschlüsselung ist es jedoch unmöglich, den Entschlüsselungs-Schlüssel aus dem Schlüssel zur Verschlüsselung zu berechnen.

In diesem Fall bezeichnet der Begriff “Netzwerk” ein System aus mehreren Rechensystemen. Nutzer greifen über eines dieser Systeme auf das Netzwerk zu.

- **Teilen:** Viele Personen können potentiell auf das Netzwerk zugreifen. Diverse Geräte können von nicht notwendigerweise gesicherten Systemen kontrolliert werden.

- **Komplexität:** Im Netzwerk laufen diverse Betriebssysteme, die über Verbindungsmechanismen miteinander kommunizieren, welche die Sicherheit gewährleisten sollten. So ein Mechanismus muss jedoch ziemlich allgemeingültig sein. Darüber hinaus kann nicht das Netzwerk in seiner Gesamtheit einer Prüfung oder einer Zertifizierung unterworfen werden.

- **Unbekannter Umfang:** Es sind nicht alle verbundenen Personen bekannt. Darüber hinaus ist nicht klar, wie andere Geräte funktionieren.

- **Menge angreifbarer Stellen:** Man kommt nicht umhin, den Sicherheitsmechanismen in allen Geräten zu vertrauen, da sich viele Netzwerkteile außerhalb der Kontrolle des Betreibers befinden.

- **Unbekannter Pfad:** In den meisten Fällen ist es nicht möglich, den Pfad der Datenübertragung zu kontrollieren. Daher kann ihn im Prinzip jeder ohne vorherige Information abfangen. Trotzdem lässt sich die Kommunikation folgendermaßen schützen:

    - Datenfluss: wird auch als “Stromverschlüsselung” bezeichnet. Darunter versteht man die Verschlüsselung von Daten auf eine Art und Weise, die den Eindruck eines sehr zuverlässigen Kommunikationskanals im Hinblick auf einen möglichen Angriff vermittelt.

    - Individuelle Nachrichten: entsprechen einer heutzutage modernen freien Bindung mittels “Messaging”. Anwendungsnachrichten werden verschlüsselt oder die Verschlüsselung des Datenstroms wird entweder zwischen zwei Netzwerkknoten oder zwei Anwendungen, welche auf diesen Netzknoten laufen, vorgenommen. Im Hinblick auf die Link-Verschlüsselung gibt es zu sagen, dass die Daten direkt vor dem Eingang in das Kommunikationsmedium verschlüsselt und direkt nach ihrer Ankunft auf dem zweiten Computer wieder entschlüsselt werden. Diese Verschlüsselung findet auf der Ebene der Bitübertragungs- oder Sicherungsschicht des Referenzmodells statt. Der Hauptvorteil ist, dass der Mechanismus für den Nutzer äußerst transparent ist. Darüber hinaus lässt er sich sehr schnell und einfach mit anderen Geräten verbinden.

Ende-zu-Ende-Verschlüsselung

Diese gewährleistet kryptografischen Schutz während des Transferprozesses. Diese Art der Verschlüsselung findet auf der Ebene der Anwendungs- oder Darstellungsschicht des Referenzmodells statt. Diese Verschlüsselung ist jedoch nicht mehr transparent und muss, um wirklich effektiv zu sein, auf geeignete Weise in das gesamte System integriert werden. Ihr Hauptvorteil besteht darin, dass es nicht nötig ist, die gesamte Kommunikation zu verschlüsseln. Es reicht, sensible Daten zu verschlüsseln. Anders als die Link-Verschlüsselung stellt diese Art Authentifizierungs- und Integritätprüfungen zur Verfügung (Ende-zu-Ende). Manchmal werden beide Methoden gleichzeitig verwendet: Die Link-Verschlüsselung für den üblichen präventiven Datenschutz und die Ende-zu-Ende-Verschlüsselung, um den qualitativ hochwertigen Schutz sensibler Daten sicherzustellen. Im Zusammenhang mit der Verschlüsselung ist Folgendes unabdingbar:

- Distributionsmechanismen
- die Verwaltung notwendiger Verschlüsselungsschlüssel,
- kompetente Instanzen zur Sicherstellung des Systembetriebs des kryptografischen Schutzes sowie geeignete kryptografische Geräte, welche grundlegende Funktionen kryptografischen Schutzes bieten.

Abgesehen von den üblichen Problemen der Zugriffsregulierung können sich noch andere heikle Angelegenheiten ergeben.

Abgestufte Zugriffsrechte

Es kann sein, dass der Zugang zu sensiblen Daten nur ein paar Netzwerkknoten vorbehalten ist. Versucht in so einem Fall, ein autorisierter Nutzer von einem anderen Netzknoten aus einen Zugang zu bekommen, können seine Zugriffsrechte massiv beschränkt werden oder der Zugang zu den Daten wird ganz verweigert. Nachdem es den Abruf erhalten hat, beginnt das stille Modem nicht sofort mit der Generierung, sondern wartet, bis die andere Seite zu „verhandeln“ versucht. Folglich ist der Zugriff bis zu einem gewissen Maß auf jene Nutzer beschränkt, welche wissen, dass man sich mit der Leitungsführung des Computers auseinandersetzen muss. Darüber hinaus beschränkt diese Methode die Möglichkeit einer willkürlichen Platzierung dieses Ports. Im Falle eines IP-Protokolls könnte ein Service, welcher auf einem bestimmten Gerät verfügbar ist, welches sich auf einem anderen Port als üblich befindet, eine Alternative zu einem stillen Modem sein.

11. Peer-to-Peer Netzwerke

In den letzten 10 Jahren gewannen Austauschnetzwerke, welche auf dem Peer-2-Peer-Prinzip basieren, im Bereich des Datei-Downloads aus dem Internet an Bedeutung. Heute ist das beliebteste das BitTorrent-Protokoll-Netzwerk, welches zig Millionen Nutzer und dutzende Clients (µTorrent, Vuze/Azureus) hat. Darüber hinaus ließ sich in den letzten fünf Jahren der Siegeszug öffentlich zugänglicher kommerzialisierter Web-Server zum Filesharing (Filehosting) beobachten, unter welchen Rapidshare.com der beliebteste ist. Zu dieser Art von Netzwerk lassen sich jedoch auch Systeme zählen, wo mehrere Nutzer an einem Dokument arbeiten oder Dateien synchronisieren (zB Google Docs, Dropbox usw.).

Peer-to-Peer Netzwerke

Der Begriff Peer-to-Peer Netzwerk (P2P) ist sehr allgemein gehalten. Darunter fallen alle Netzwerke, in welchen es eine symmetrische Kommunikation oder Interaktion zwischen Computern gibt (jeder davon kann diese selbst initiieren oder auf Grundlage einer externen Initiierung/Aufforderung die notwendigen Transaktionen und Operationen durchführen). Deshalb ist es nicht möglich, seine Netzknoten funktionell in die zwei üblichen Arten zu unterteilen, sprich in Clients und Server. Dementsprechend fehlt eine grundlegende Aufgabe des Servers. Daraus folgt, dass eine hybride, universelle Art von Computernetzknoten gebildet wird, welcher „Servent“ genannt wird (auch wenn der Begriff „Client“ häufig dafür verwendet wird). Die Dezentralisierung des Internets, welche im Zeitalter von Usenet (1979) und Fidonet (1984, BBS Systeme) begann, wurde letztlich durch P2P-Netzwerke erreicht. Die Gleichwertigkeit aller Computer in diesem Netzwerk bedeutet jedoch nicht, dass alle Netzknoten dieses Netzwerks homogen sind, was die quantitativen Parameter wie Verbindungs-geschwindigkeit, Menge geteilter Daten, lokale Konfiguration, Verarbeitungsgeschwindigkeit usw. anbelangt.

Im Großen und Ganzen ist die Marke P2P mittlerweile fast zu einem Gattungsnamen für internetbasierte Austauschnetzwerke geworden (sprich für Filesharing-Systeme), obwohl es eine Reihe anderer Anwendungsarten gibt, welche nach demselben Prinzip funktionieren können, zB Anwendungen für dezentralisierte Berechnungen, die Nachrichtenverbreitung, Sprachkommunikation und Chat (IRC, Skype, Qnext, DKMessenger) oder P2P-Übertragung des Programms von Radio- und TV-Stationen im Internet.

Nun ist es jedoch so, dass beliebte P2P-Netzwerke (Filesharer) von den Medien zumeist mit Piraterie in Verbindung gebracht werden. Dies mag zum Teil auch stimmen. Dabei darf man jedoch nicht vergessen, dass zB das BitTorrent-Netzwerk auch stark dafür genutzt wird, große Programme und Datenmengen auf legale Weise zu verbreiten (zB LINUX). Darüber hinaus gibt es Torrent-Kataloge und –Tracker, welche dabei helfen, ausschließlich legales Material (zB Filme) zu verbreiten und zu registrieren – und zwar in gemeinfreier Qualität (http://www.legittorrents.info/, http://beta.legaltorrents.com/, http://www.jamendo.com/, http://www.publicdomaintorrents.com/,); sprich welche aus

rechtlicher Sicht kostenfrei genutzt werden dürfen.
Ein Unterscheidungsmerkmal von P2P-Austauschnetzwerken ist, dass alle Dateien oder Text-nachrichten, die auf einem Computer gespeichert sind, welcher mit dem Internet verbunden ist (und mit einem laufenden Client/Server eines bestimmten Netzwerks ausgestattet ist), innerhalb dieser Netzwerke geteilt werden können. Auf der anderen Seite unterscheiden sich die Netzwerke technisch in Bezug auf den tatsächlichen Grad funktioneller Symmetrie, sprich ihre Homogenität und Dezentralisierung und ihrer relativen „Anonymität". Es ist so, dass es so etwas wie vollständige Anonymität im Internet praktisch nicht gibt. Verschiedene Generationen dieser Netzwerke unterscheiden sich nach den oben genannten Merkmalen.

11.1. Generationen von P2P-Netzwerken

Zur ersten Generationen gehören Netzwerke, welche Datei-/Computerlisten sowie Datei- und Computeradressen auf einem oder mehreren zentralen Spezialservern speicherten (Napster, OpenNap, chat – IRC [seit 1988]/IRC@find, Soulseek – das bislang neueste Werk, welches von einer kleineren Community von Musik-Fans genutzt wird). In Bezug auf deren weitere Entwicklung lässt sich sagen, dass die ältesten P2P-Netzwerke für einige ihrer Aktivitäten eine zentralisierte Struktur der Art Client/Server nutzen. Diese beschäftigt sich vor allem mit der Netzwerkverbindung und der Suche nach Ressourcen (Dateien). Aus diesem Grund braucht man keine Server für das routinemäßige Kommunizieren und Teilen zwischen Ende-zu-Ende-Clients. Solche Aufgaben übernehmen Peer-2-Peer-Netzwerke.

Netzwerke der zweiten Generation sind heutzutage die am häufigsten verwendeten. Zentrale Server fehlen komplett: Gleichwohl basieren diese Netzwerke in der Regel nicht auf vollkommen gleichwertigen Servent-Netzknoten, sprich Computer können nicht im großen Stil durch andere Computer ersetzt werden (nur bei Vertretern wie Gnutella oder Freenet). In der Tat wird die perfekte Symmetrie in den Peer-to-Peer-Regeln in der Praxis oft reduziert, um die Effektivität und Zuverlässigkeit der Suche zu erhöhen oder das Identifizieren von Dateien zu erleichtern und somit deren Downloadgeschwindigkeit zu erhöhen.

Heutige P2P-Systeme sind durch "lokale-zentrale" oder speziell abgestellte/zweckbestimmte Komponenten gekennzeichnet, wie zB verschiedene "Super-Netzknoten" (FastTrack-Netzwerk mit Clients wie Kazaa), Netzknoten (DirectConnect, DC++) oder Suchmaschinen und Indexierer (OpenFT), aus denen alle (auf freiwilliger Basis) einzelne Computer werden können. Diese speziellen „Super-Netzknoten" sind wichtig für deren schnelle Verbindung und das Katalogisieren aller anderen gewöhnlichen Computer-Knoten und ihrer Ressourcen. Sie verhindern auch das Auftreten von „Flaschenhälsen" innerhalb von Datenströmen. Die weiteren "asymmetrischen" Komponenten bezeichnen spezialisierte Speicher-Server, welche digitale Hash-Abbilder einzelner

Dateien bereitstellen (Identifizierungssignatur, zB Torrents wie bei BitTorrent, Magnetstreifen wie bei Gnutella, ed2k wie bei eDonkey/Overnet-Leitungen). Es handelt sich hierbei um Server, welche dabei helfen, ein beschleunigtes „schwimmendes“ Herunterladen von einzelnen Teildatensegmenten zwischen Computern zu koordinieren.

Trotzdem nutzen neuere Netzwerke die Peer-to-Peer-Struktur für so ziemlich jeden Zweck bzw. jede Aufgabe. Aus diesem Grund werden sie auch „echte P2P-Netzwerke“ genannt. Darüber hinaus haben echte P2P-Netzwerke das unverwechselbare Merkmal, dass sich die gesamte Übermittlungs- oder Kommunikationsfähigkeit eines gewöhnlichen Nutzers mit der Anzahl der Nutzer/Netzknoten erhöht. Im Gegensatz zur verringernden Natur zentralisierter Systeme wird diese Tatsache durch die Technologie des Segment-Downloads unterstützt.

Ein weiteres wesentliches Feature der zweiten Generation von P2P-Netzwerken sind Multiprotokoll-Clients, welche in der Lage sind, innerhalb mehrerer Netzwerke (Protokolle) nach Daten zu suchen und diese herunterzuladen – manchmal sogar gleichzeitig. Hierzu gehören zB MLDonkey, KCeasy und Shareaza. Darüber hinaus sind sogenannte „Overlay-Protokolle“ ein integraler Bestandteil, welche eine Art „Supernetzwerk“ innerhalb eines bestimmten Netzwerks (Overnet) bilden.

Mit dem Aufkommen der dritten Generation von Netzwerken und Clients einher ging die Einführung bzw. die Verbesserung der Verschlüsselungselemente (Verschleierung des Datenverkehrs, welcher zum BitTorrent-Netzwerk hinzugefügt wurde), Anonymität oder Pseudo-Anonymität, sprich das verheimlichen der IP-Adressen von Computern/Netzknoten (Verschlüsselung des beidseitigen Routings oder Verknüpfens). Darüber hinaus versucht sie, eine größere Dezentralisierung zu erreichen, als das heute der Fall ist (Freenet, GNUnet). Tatsächlich kann man auf einige dieser Netzwerke nicht ohne die Zustimmung von anderen Nutzern zugreifen (Netzwerke wie Freund-zu-Freund, zB ANts P2P, WASTE, MUTE). Andere wiederum haben beinahe die Merkmale eines Virtual Private Network (VPN), zB I2P, welches das private Ausführen aller Internetaktivitäten erlaubt. Manchmal bieten solche Netzwerke nicht bloß Filesharing-Dienste an, sondern auch einen geschützten Bereich für die Kommunikation und Veröffentlichung von Dokumenten, ohne dass diese durch Zensur beeinträchtigt werden. Somit werden sie dezentralisiert und von einer privaten Groupware geschützt. Auf der anderen Seite sind die größten Nachteile dieser Systeme der geringere Nutzungskomfort sowie deren geringe Geschwindigkeit.

Die oben angesprochenen Internet-Radios und TV-Übertragungen (Internet-Streaming) des Typs Peer-to-Peer gehören manchmal zu den P2P-Anwendungen der vierten Generation (TVUPlayer, PPLive, PeerCast, PPStream). Ein bestimmter Teil von P2P-TV-Systemen basiert auf dem BitTorrent-Protokoll oder etwas ähnlichem (in diesem Fall geht es jedoch um Fernsehen „auf Abruf“, kein Live-Streaming). Gleichzeitig nutzen die größten davon ihre eigenen, unterschiedlichen Systeme. Alles in allem werden dutzende, wenn nicht gar hunderte von TV-Kanälen (oft Sport) auf diesem Wege übertragen –

und zwar mit hoher Effizienz (vorausgesetzt, jeder empfangende Netzknoten kann zugleich auch ein rücküber-setzender Transmitter für andere Netzknoten sein.)

11.2. Filehosting

Dann gibt es da noch das Filehosting (sowohl frei als auch kommerziell) auf Web-Servern, einen im Wesentlichen ziemlich konservativen und technisch alten Dienst, der in Anbetracht der heutigen Verbindungsgeschwindigkeit jedoch sehr effektiv ist. In diesem Bereich gibt es über 100 wichtige ausländische Server, darunter auch so um die 10 tschechische. Die Voraussetzungen und Systeme für ihre Nutzung variieren je nach Server im Hinblick auf das Up- und Downloaden. Sie haben jedoch oft zwei verschiedene Download-Modi, sprich kostenlose Downloads können entweder nach Volumen oder Zeit eingeschränkt sein (finanziert durch Werbung und begrenzt durch OCR/Captcha-Systeme, temporäre Leitungen und Zeitintervalle) und "Bonus-Downloads", welche gegen Bezahlung einer kleinen Gebühr kaum bzw. uneingeschränkt zur Verfügung stehen. Eine begrenzte Liste dieser Server, ohne die dazugehörigen Domain-Endungen, welche sich mit etwas Recherche selbst weiter fortsetzen lässt, ist nachfolgend zu finden:

Im Ausland:
Rapidshare, depositfiles, filefactory, megaupload, mediafire, sendspace, uploading, zshare, ifolder, hotfile, icefile, letitbit, filefront, ifile, easy-share.

In Tschechien: Edisk, uloz.to, czshare, leteckaposta, quickshare, bagruj, nahraj.
In diesem Bereich gibt es ein paar Programme, welche bis zu einem gewissen Grad das Downloaden von diesen Servern automatisieren und somit erleichtern können. Dazu zählen zB der Universal Share Downloader (USD) oder RapGet. Andere wichtige Links sind:

- www.filesharing.eu Filesharing über P2P-Netzwerke
- www.slyck.com Nachrichtenplattform
- www.zeropaid.com Nachrichtenplattform
- www.filesharingz.com Nachrichten
- www.p2pnet.net Nachrichten
- www.p2pforums.com Diskussions- und Nachrichtenplattform
- www.infoanarchy.org Dezentralisiertes P2P-Wikiportal
- www.openp2p.com O´Reillys Website mit Fokus auf den P2P-Bereich
- www.planetpeer.de Plattform für anonyme Netzwerke, nicht nur für Filesharing (MUTE, I2P, Freenet etc.)
- www.fileshareworld.com Wegweiser für Austauschsysteme und P2P-Clients
- www.ftc.gov/bcp/workshops/filesharing Meinungen der US-Handelskommission über Filesharing
- http://cs.wikipedia.org/wiki/Peer-to-peer Tschechisches Wikipedia-Passwort
- http://p2ptv.yourglobaltv.com/, http://www.tvavailable.com/P2P_TV/ Überblick

über die häufigsen P2P-TV-Clients

- http://en.wikipedia.org/wiki/P2PTV Augenöffnender Artikel über P2P-TV
- http://www.yourglobaltv.com/p2pchannels/ Hauptsportkanäle von P2P-TV
- http://en.wikipedia.org/wiki/File_hosting_service Allgemeines über Hosting-Server
- http://www.dimonius.ru/dusd.php automatischer Downloader Universal Share Downloader (USD)
- http://www.rapget.com/en/ RapGet ähnlich dem oben genannten USD.

12. Anonymität im Internet

Die Anonymität im Internet war in letzter ein heiß diskutiertes Thema, sogar in den Regierungen hoch entwickelter Länder. Anonymität war schon immer ein wichtiger Faktor und ist es auch heute noch. Nichtsdestotrotz ist es aufgrund des Aufkommens neuer Technologien und ihres damit einhergehenden Siegeszugs in der Gesellschaft notwendig geworden, sich intensiver damit auseinanderzusetzen als bisher. Auf der anderen Seite haben einzelne Länder bereits erste Schritte eingeleitet, um das Internet einer stärkeren Kontrolle zu unterziehen. Nutzer sollen dadurch besser identifizierbar sein, womit die Sicherheit erhöht werden soll (dies zielt hauptsächlich auf die Bekämpfung von Terrorismus, Kinderpornografie, Drogenhandel und Geldwäsche ab). Gleichwohl versuchen Länder jedoch auch, die Privatsphäre und sensible persönliche Daten von Internetnutzern zu schützen, was von der EU mit der Durchsetzung eines sogenannten „Cookie-Gesetzes“ unterstützt wurde.

Um die soziale Anonymität zu gewährleisten, müssen alle sieben Dimensionen von personenbezogenen Daten ausgeklammert werden:

- Vollständiger Name
- Wohnort
- Pseudonym, welches mit dem echten Namen oder dem Wohnort in Verbindung steht
- Pseudonym, welches auf andere persönliche Informationen schließen lässt
- Offenlegen von Verhaltensmustern
- Mitgliedschaft in einer bestimmten sozialen Gruppe
- Informationen, Objekte oder Fähigkeiten, welche auf persönliche Eigenschaften schließen lassen.

Die Gründe für die Wahrung der Anonymität bleiben in beiden Welten – sowohl der realen als auch der virtuellen – gleich: Das Vermeiden von Konsequenzen, die sich aus dem eigenen Verhalten ergeben (Furcht vor Repressionen, Missverständnissen usw.)

Identifizierungstechnologien

Moderne Technologien bieten eine Reihe von Wegen und Methoden zur Identifizierung von Individuen an.

IP-Adresse

Diese bezieht sich im Wesentlichen auf die eindeutige Adresse jedes Geräts, welches mit dem Netzwerk verbunden ist. Eine IP-Adresse kann zB Informationen über den geografischen Standort preisgeben.

Geolokalisierung

In letzter Zeit ist die Nutzung von Technologien, welche in der Lage sind, den Standort von Personen auf den Meter genau zu ermitteln, stark erhöht. Beispiele hierfür sind:

- Einschränkungsbasierte Geolokalisierung: Diese setzt auf das aktive Messen der Distanz über eine Antwort.

- GPS läuft hauptsächlich auf Mobiltelefonen.

- Eine Inhaltsanalyse von Web-Postings (Sozialen Netzwerken): Dies bezieht sich auf eine Methode der Lokalisierung von Nutzern auf Grundlage der Analyse von öffentlich verfügbaren Postings, zB nach der Stadt, dem Land oder dem Wetter, welche bzw. welches im Inhalt des Postings zu finden ist. Darüber hinaus lässt die Aktivität eines Nutzers zu einer bestimmten Zeit einen Schluss auf die Zeitzone zu, in der er sich befindet.

Cookies

Cookies dienen als ständiger Identifikator zwischen einem Client und einem Server. Durch ihr Eindringen in die Privatsphäre der Nutzer wird mit Cookies innerhalb der Europäischen Union jedoch sehr kritisch umgegangen.

PRISM

PRISM bezieht sich auf ein staatliches Projekt des US-amerikanischen Sicherheitsdienstes NSA. Dieses Projekt hat uneingeschränkten Zugriff auf Daten von Google, Microsoft, Yahoo und viele andere Dienste, welche mit Blick auf die oben genannten Unternehmen 98% aller Daten erzeugen. Trotzdem leugnen besagte Unternehmen, dass sie unbeschränkten Zugang zu all diesen Daten haben.

12.1. Technologien zum Schutz der Privatsphäre

Tor bezeichnet ein Projekt, welches auf dem Onion Routing-Konzept basiert und damit die Anonymität der Nutzer sicherstellt, während diese im Internet surfen. Das Programm ist darauf ausgelegt, die personenbezogenen Daten der Nutzer, die Privatsphäre sowie die Möglichkeit auf den geheimen Abschluss von Geschäften zu schützen, indem es die Nutzer davor bewahrt, dass ihre Aktivitäten im Internet nachverfolgt werden. Eben diese Software lässt sich jedoch auch für illegale Aktivitäten missbrauchen, da die Verwendung dieser Technologie das Ausforschen von Kriminellen erschwert.

Orbot bezeichnet die Tor-Version für Mobiltelefone mit dem Betriebssystem Android.

Freenet-Projekt wird als eine vollkommen sichere, anonyme und dezentralisierte Plattform zum Teilen von Daten betrachtet, deren Zweck darin besteht, die Zensur auszuhebeln. Jeder Nutzer stellt einen Teil seiner lokalen Festplatte zur Verfügung, auf welchen das gesamte Netzwerk zugreifen kann. Darauf können verschlüsselte Daten von anderen Nutzern gespeichert werden. Dadurch wird kein eigener privater Server benötigt.

Das Internet umfasst ein ausgedehntes und komplexes Netzwerk von Kommunikationskanälen, über welche Daten fließen. Jedes **Datenpaket** (in der Folge nur noch "Paket" genannt) enthält – abgesehen vom Inhalt selbst, welcher irgendwo hingeschickt werden soll – bestimmte Erkennungsmerkmale, welche von besonderem Interesse für neugierige Behörden und Unternehmen sind. Dazu gehören **Google, Microsoft, Facebook, Twitter** usw.

In der Tat nutzen diese Firmen **Erkennungsmerkmale** von Nutzerdaten, um herauszufinden, welche Webseiten für sie von Interesse sind, wie oft sie diese besuchen und auch, von wo aus sie darauf zugreifen. In der Folge werden diese wertvollen Informationen ausgewertet, um zielgerichtete Werbeanzeigen schalten zu können. Seit dem Aufkommen sozialer Netzwerke geht diese Spionage noch deutlich weiter.

Die großen Behörden und Unternehmen, welche auch "Big Brother" genannt werden, können die Online-Aktivitäten von Nutzern erfolgreich mit jenen ihrer Freunde **kombinieren** und somit eine genaue Karte der Internetnutzer zeichnen. Dieser **freiwillige Verzicht auf die Privatsphäre** ist der Preis für die kostenlose Bereitstellung von Diensten. Auf der anderen Seite, um nicht nur die Nachteile von diesen Serviceanbietern aufzuzeigen, ist es notwendig darauf hinzuweisen, dass ihre Dienste qualitativ sehr hochwertig und durchdacht sind, auch wenn dies auf den ersten Blick nicht so scheinen mag.

12.2. Grundprinzipien zur Wahrung der Anonymität

Wie bereits gesagt wurde, verfolgt das Internet hauptsächlich, woher unsere Daten kommen und ob sie zu uns gehören. Daher besteht die Hauptaufgabe darin, diese **Daten zu verschleiern**. Zu diesem Zweck sollte man auf VPNs (Virtual Private Networks), Proxy-Server oder spezielle Internetbrowser setzen. Die Vor- und Nachteile der einzelnen Lösungen werden nachfolgend vorgestellt.

Verbindung via VPN

Virtuelle private Netzwerke werden vor allem in Unternehmen verwendet, wo sie es mehreren Computern ermöglichen, auf Netzwerkspeicher, Drucker oder andere Server zuzugreifen. Gleichzeitig stellen sie eine Kommunikation zum frei zugänglichen Internet her. Des Weiteren werden Datenpakete mit der IP-Adresse eines VPN-Servers markiert, nicht mit jener des Computers, damit Big Brother sie diesem nicht zuordnen kann.

Loggt man sich in ein VPN ein, welches sich in einem anderen Land befindet, kann der physische Standort nicht herausgefunden werden. Darüber hinaus verschlüsselt ein VPN-Server die Kommunikation, damit abgesehen vom Ende-zu-Ende-Server niemand in der Lage ist, die Daten auszulesen.

Es gibt viele VPNs aus den verschiedensten Orten der Welt, welche für das anonyme Surfen im Internet bestimmt sind. Leider sind sie verpflichtet, die IP-Adresse von Nutzern offenzu-legen, wenn sie von einem Gericht dazu aufgefordert werden: Anderenfalls würden sie verboten werden. Deshalb kann man sich nicht darauf verlassen, dass niemand dazu in der Lage ist, die IP-Adresse eines Nutzers herauszufinden, nur weil er einen VPN-Dienst nutzt.

Verbindung via Proxy-Server

Ebenso wie ein VPN-Server stellt auch ein Proxy-Server die Kommunikation zwischen einem PC und dem Internet her. In diesem Fall wird das Internet Nutzerdaten so betrachten, als gehörten sie zum Proxy-Server. Anders als bei einem VPN werden die Daten jedoch nicht verschlüsselt. Da Proxy-Server eine einfachere Lösung als VPNs darstellen, ist es auch möglich, Proxy-Server für öffentlich zugängliche Online-Schnittstellen zu verwenden.

In der Tat lassen sie sich leicht einsetzen, wenn es um eine kurzfristige Anwendung geht. Man muss nur die richtige Adresse in ihre Adresszeile eintragen. Dadurch stellt der Proxy-Server eine Kommunikation mit dem bestimmten Web her.

Gleichwohl muss man beachten, dass alle Nutzerdaten über den Proxy-Server laufen, sprich im Falle eines unzuverlässigen Proxy-Servers können Daten missbraucht werden. Aus diesem Grund wird davon abgeraten, Proxy-Server für den Besuch geschützter Webseiten (https) zu benutzen.

Spezielle Web-Browser und Suchmaschinen

Anonyme Web-Browser konzentrieren sich auf Layman's Erwartungen an seinen üblichen Browser, wenn der Anonymitätsmodus eingeschaltet ist. Dieser Modus versucht nicht um jeden Preis, die Internetaktivitäten eines Nutzers vor Big Brother zu schützen – sondern nur vor anderen Nutzern desselben Computers. Somit wird der gesamte Internetverkehr wie üblich nachverfolgt.

Im Gegensatz dazu sind anonyme Web-Browser so programmiert, dass sie Big Brother daran hinder, Nutzerdaten auszuspähen. Auch wenn diese Lösung nicht außergewöhnlich effektiv ist, so ist sie zumindest einfach und schnell. Darüber hinaus ist es sehr angenehm, einen Browser mit einer anonymen Internetsuchmaschine statt Google, Yahoo oder Bing zu verwenden, da dieser Browser keine Informationen darüber speichert, wo-

nach der Nutzer gesucht hat.

Onion Routing – Tor

Tor ist ein Web-Browser und auch ein Server-Netzwerk. Es wird oft als Internet im Internet bezeichnet. Es arbeitet nach dem Prinzip des Onion Routings, demzufolge jedes Paket in Schichten verschlüsselt ist – wie eine Zwiebel – und jeder Server nur dazu in der Lage ist, eine Schicht zu dechiffrieren.

Dadurch durchläuft jedes Paket eine bestimmte Anzahl von Servern (zumindest drei), bevor es sein Ziel erreicht. Nichtsdestotrotz kennt jeder Tor-Server nur die Adressen des Ankunfts- und Ausgangsservers, nicht jedoch die gesamte Kaskade.

Deshalb ist es fast unmöglich, ein Paket bis zu seiner ursprünglichen IP-Adresse zurückzuverfolgen. Tor hat auch dadurch einen zweifelhaften Ruf erworben, dass es einen Zugang zum Darknet erlaubt, in welchem allerlei illegale Aktivitäten stattfinden. Der größte Nachteil beim Einsatz eines Tor-Browsers ist dessen geringe Geschwindigkeit, da die darüber transferierten Daten buchstäblich in und quer durch die gesamte Welt reisen.

12.3. TOR: Total anonymes Internet

Digitaler Fußabdruck

Jeder Nutzer des weltumspannenden Netzwerks hinterlässt Spuren seiner Aktivitäten. Dieses Phänomen wird seit den 1980ern als „digitaler Fußabdruck“ bezeichnet. Zu einem gewissen Grad hängt die Intensität des Fußabdrucks – zumindest in der ersten Phase – vom Nutzer ab: Dies gilt vor allem für die Browser-Einstellungen oder die Installation des Proxy-Servers, welcher die Kommunikation zwischen dem internen Netzwerk und dem Internet bereitstellt und somit eine Reihe von Sicherheitseinstellung ermöglicht. Alles in allem garantiert er keine vollständige Anonymität, gewährleistet jedoch den adäquaten Schutz der Privatsphäre für die übliche Nutzung des Internets.

Ein effektiverer Schutz

Zu Beginn des Jahrhunderts wurde mit Argusaugen auf die Anonymität im Internet geachtet. Ein paar Projekte wurden ins Leben gerufen, um Nutzer mit einem geeigneten Schutz ihrer Privatsphäre auszustatten. Besondere Aufmerksamkeit kam dem Projekt des Tor-Netzwerks zu (Onion-Routing), da es von der US-amerikanischen Regierungsbehörde Naval Research Laboratory gegründet und finanziert wurde. Zunächst diente es zum Schutz der Regierungs-kommunikation in den USA. Eine weitere finanzielle Förderung erhielt Tor von der gemein-nützigen Organisation Electronic Frontier Foundation, welche sich mit dem Schutz von Rechten in der digitalen Sphäre beschäftigt.

Knoten neben dem Knoten

Wie funktioniert ein vollständig anonymes Netzwerk? Das Sicherheitsprinzip basiert auf einer großen Anzahl an Knoten, welche der Datenstrom passiert. Die gesamte Kommunikation wird asymmetrisch verschlüsselt und Server teilen öffentliche Schlüssel und tauschen erzeugte dynamische Schlüssel aus um untereinander Datenstrings zu übermitteln. Jeder Knoten ist völlig autonom und zieht die Information darüber, wohin der String zu senden ist, nur aus dem Nachrichtenkopf. Damit endet sein Wissen über den Datenfluss beim nächsten Knoten. Um es einfacher auszudrücken: Der Server weiß, an wen eine bestimmte Information geschickt werden soll und an wen die Antwort zurückgehen soll. Andere Aktivitäten übersteigen jedoch seine Fähigkeiten. Je mehr Knoten eingebunden sind, desto höher ist die Anonymität. Der Netzwerknutzer selbst wählt einen geeigneten Pfad, über welchen die Daten geschickt werden sollen.

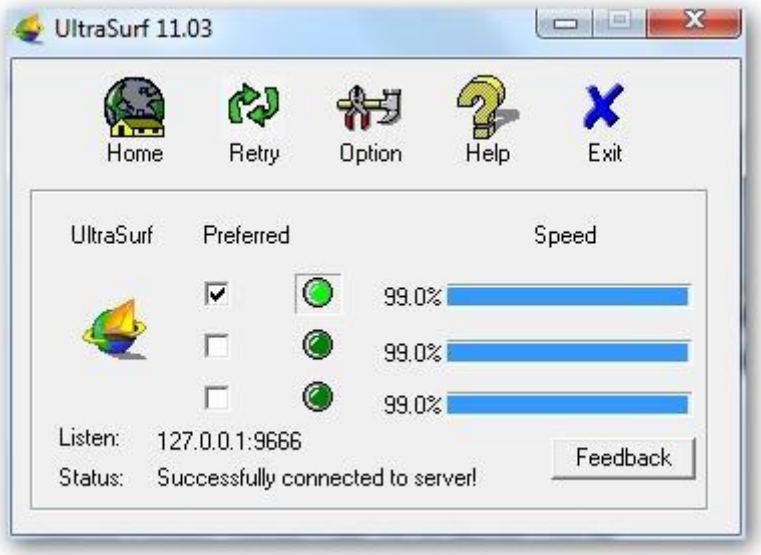
Falsche Fußabdrücke

Die Komplexität des Datentransfers über anonyme Netzwerke scheint auf den ersten Blick sehr zufriedenstellend zu sein: Tor geht jedoch noch weiter. So können seine Server Pakete zB in einer anderen Reihenfolge verschicken, als sie empfangen werden. Ebenso können falsche Pakete können zwischen den übermittelten Datenstrings eingeführt werden, welche eine mögliche Nachverfolgung des Datenstroms vom tatsächlichen Pfad umlenken. Der Hauptnachteil des Tor-Netzwerks ist dessen lange Reaktionszeit.

Darüber hinaus gibt es Situationen, in welchen ein Nutzer, um seine Privatsphäre zu schützen oder auf einige Dienste zuzugreifen, seine IP-Adresse ändern muss. Dies ist trotzdem eine heikle Operation: Viele Dienste bieten eine Lösung über ein VPN und ihren eigenen Client mit Extra-Funktionen an, einen Zugriff auf ihre eigenen Server, wodurch die Kommunikation so umgeleitet wird, dass der Nutzer entweder anonym oder verschlüsselt im Internet surfen kann.

UltraSurf

UltraSurf ist ein minimalistisches Programm, welches selbst absoluten Anfängern anonymes Surfen im Internet erlaubt. Erreicht wird dies vor allem durch einfache Einstellungen, Bedienung, günstigen Betrieb und freie Verfügbarkeit. Das Programm im kleinen Fenster bietet eine Verbindung zu drei verschiedenen Servern an.



Die erfolgreiche Änderung der IP-Adresse wird von einem Icon in Form eines Vorhängeschlosses in der Toolbar angezeigt. Darüber hinaus ist es möglich, die Bedienungselemente mithilfe von Tastenkombinationen ein- bzw. auszuschalten, den Browser zu starten, automatisch Cookies und Verlauf zu löschen sowie den Proxy-Server einzurichten. Um auf US-amerikanische Server und Dienste zuzugreifen, welche eine locale IP-Adresse benötigen, nutzt UltraSurf den “freekarol”-Reader, womit es zu den besten kostenlosen Anwendungen gehört.

proXPN

Dieser VPN-Client ist sehr beliebt und wird oft verwendet, da er eine gute Leistung, qualitativ hochwertige Dienste, die Wahl zwischen einer US-amerikanischen, britischen, holländischen und singapurischen IP-Adresse, Verschlüsselung, unbegrenzten Datentransfer, zeitlich begrenzte Informationsspeicherung über die Verbindung (zwei Wochen) und eine Reihe weiterer Features anbietet. Das Programm umfasst in der kostenlosen Version die obengenannten Dienste, weitere Features gibt es gegen Gebühr.

TunnelBear VPN

Benutzer bezeichnen den Dienst TunnelBear und seine VPN-Anwendung als schön, sehr einfach und nutzerfreundlich. Darüber hinaus ist es möglich, eine kostenlose Version zu verwenden, welche es erlaubt, 500 MB Datenvolumen pro Monat kostenlos durch den „Tunnel“ zu schleusen. Darüber hinaus gibt es 1 GB extra, wenn man ihn auf Twitter empfiehlt. Gegen eine Gebühr von fünf US-Dollar pro Monat bekommt man die Möglichkeit, auf die unlimitierte Version zuzugreifen.

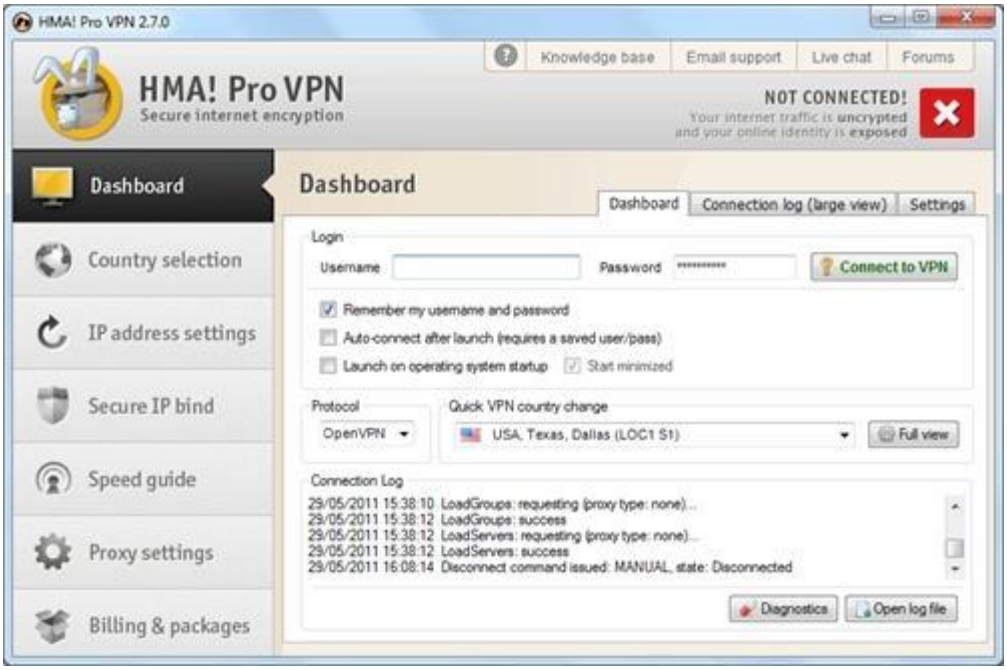
CyberGhost VPN

Ein weiteres empfohlenes Tool ist CyberGhost, welches eine verschlüsselte Verbindung (1024 Bit SSL 128 Bit AES-Passwort) zu einem VPN über einen Software-Client voraussetzt, über welchen die weitere Kommunikation mit dem Internet geführt wird. Das Programm ist leicht zu bedienen und bereits mit wenigen Klicks ist eine Verbindung hergestellt.

Das Server-Netzwerk von CyberGhost ist beständig und effizient. Darüber hinaus ist der Geschwindigkeitsverlust im Vergleich zu anderen Lösungen äußerst gering. Das Programm bzw. der Dienst können kostenlos genutzt werden, wenn auch nur mit funktionellen Einschränkungen (1 GB Datenvolumen pro Monat mit einer Geschwindigkeit von 2 MBit/s usw.). Darüber hinaus bieten die Hersteller verschiedene bezahlte und nach Effizienz gestaffelte Versionen an. CyberGhost wurde von Suslikus empfohlen.

Hide My Ass

Hide My Ass ist ein weltbekanntes VPN, welches eine breite Nutzerbasis hat und über ein breites Spektrum an Diensten und Servern verfügt. Hide My Ass hat fast 40.000 IP-Adressen in 53 verschiedenen Ländern im Angebot, hinter welchen man sich verstecken kann. In der Tat können kostenlose Lösungen nicht mit so einer großen Zahl an IP-Adressen mithalten, dafür fallen jedoch auch mehr als 11 US-Dollar pro Monat an Nutzungsgebühr an. Wenn man sich für ein langfristiges Abonnement entscheidet, reduziert sich der Preis jedoch auf sechseinhalb US-Dollar.



13. Angriff und Verteidigung im Internet

13.1. Angriff und Verteidigung des PCs

13.1.1. ANGRIFF

Virus

Der Name leitet sich aus der Analogie zum Virus biologischen Ursprungs ab. Er kann sich selbst reproduzieren – vorausgesetzt, es gibt einen funktionstüchtigen Wirt, mit welchem er verbunden ist. Ein Wirt kann aus ausführbaren Dateien, Systemplattenbereichen oder Dateien bestehen, welche nicht direkt ausgeführt werden können, aber über den Umweg einer bestimmten Anwendung (Microsoft Word-Dokumente, Visual Basic Scripts usw.). Wird der Wirt gestartet, wird zeitgleich der Virus-Code dechiffriert. Zu diesem Zeitpunkt versucht der Virus für gewöhnlich, eine weitere Selbst-Reproduktion sicherzustellen – und zwar durch das Herstellen einer Verbindung zu anderen geeigneten lauffähigen Wirten.

Trojaner

Anders als ein Virus kann dieser Schadcode sich nicht selbst reproduzieren und Dateien infizieren. Ein Trojaner scheint meist eine ausführbare EXE-Datei zu sein, welche nichts (Nützliches) enthält, sondern nur den Trojaner. Sobald er aktiviert wurde, gibt es – da der Trojaner sich nicht mit einem Wirt verbindet – nur eine Möglichkeit der Desinfektion: Die Löschung der infizierten Datei. Eine ältere Definition besagt, dass obwohl ein Trojaner nützlich zu sein scheint, er in Wahrheit ziemlich schädlich ist. Vor langer Zeit tauchte ein Trojaner auf, der wie ein Virusscanner von McAfee aussah, jedoch nichts anderes Tat, als Daten auf der Festplatte zu entfernen. Aktuell stößt man mitunter auf folgende Arten:

- passwortstehlender Trojaner (PWS)
- zerstörerischer Trojaner
- Backdoor-Trojaner
- Proxy-Trojaner

Wurm

Ursprünglich bezog sich der Begriff „Wurm“ auf „Morris Wurm“, welcher 1989 einen beachtlichen Teil des damaligen Netzwerks befiehl, welches sich später zum Internet entwickelte. Dieser und andere Würmer (die aktuellsten sind Code Red, SQL Slammer, Lovsan/Blaster, Sasser usw.) operieren auf einer niedrigeren Netzwerkebene als gewöhnli-

che Viren. Sie verbreiten sich nicht in Form von befallenen Dateien, sondern als Netzwerkpakete. Diese Pakete werden von einem erfolgreich infizierten System an andere Internetsysteme gesendet (entweder zufällig oder über einen bestimmten Schlüssel). Erreicht solch ein Paket ein System mit einer bestimmten Sicherheitslücke, kann dieses infiziert werden, wodurch in der Folge weitere „Wurmpakete“ produziert werden können. Folglich verbreiten sich Würmer, indem sie bestimmte Sicherheitslücken des Betriebs-systems ausnutzen. Der Erfolg von Würmern hängt von der Verbreitung der Software ab, welche ausnutzbare Sicherheitslücken hat. Wie sich aus den oben genannten Charakteristika vielleicht ableiten lässt, können Würmer nicht von einer herkömmlichen Antivirensoftware entdeckt werden. Daraus folgt, dass als Nebeneffekt eine ernsthafte Netzwerkinfektion auftreten kann, welche auch ein Unternehmens-LAN betreffen kann. Der Begriff „Wurm“ wird oft mit einer Infiltrationsart assoziiert, welche per E-Mail verbreitet wird. In diesem Fall überschneiden sich die Begriffe „Wurm“ und „Virus“.

Spyware

Der Ausdruck Spyware bezieht sich auf ein Programm, welches das Internet nutzt, um ohne das Wissen eines Nutzers Daten von einem Computer zu verschicken. Anders als ein Backdoor-Trojaner stiehlt diese Schadsoftware nur statistische Daten wie eine Aufstellung der besuchten Webseiten oder installierten Programme. Dies wird gemacht, um die Bedürfnisse und Interessen von Nutzern herauszufinden und diese Informationen für gezielte Werbung zu nutzen. Es gibt jedoch keine Garantie, dass sich diese Informationen oder Technologie nicht auch missbrauchen lassen. Deshalb gibt es eine Menge von Nutzern, welche über die bloße Existenz und Legalität von Spyware erzürnt sind. Dazu mag sicher auch kommen, dass Spyware häufig mit ausdrücklicher Zustimmung ihrer Verfasser mit einer großen Zahl von Shareware-Programmen verbreitet wird.

Adware

Diese bezieht sich auf ein Produkt, welche durch Werbung am PC die Arbeit mit diesem erschwert. Die typischen Symptome sind Pop-Up-Werbefenster beim Surfen und aufdringliche Webseiten (zB Startseiten im Internet Explorer), welche für den Nutzer uninteressant sind. Oft ist eine „EULA“, was Endbenutzer-Lizenzvertrag bedeutet, Teil dieser Adware, aufgrund welcher der Nutzer in vielen Fällen der Installation zustimmen muss. Adware kann Teil eines bestimmten Produkts (zB BSPlayer) sein. Im Gegenzug dafür, dass man die ständige Werbeeinblendung akzeptiert, bekommt man als Belohnung eine größere Zahl nützlicher Funktionen, welche eine gemeinhin kostenfreie Version (werbefrei) nicht bietet.

Dialler

Ein Dialler ist ein Program, welches die Art des Internetzugangs via Modem verändert. Anstelle der üblichen Telefonnummer für die die Internetverbindung leitet das Wählprogramm das Modem auf eine Mehrwertnummer um, zB 60 Kronen/Minute. Man muss

jedoch anmerken, dass diese Dialler nur in Zeiten der analogen Telefonleitungen (Ein-Wähl-Verbindungen) effektiv waren und ADSL oder andere moderne Technologien nicht davon betroffen sind.

Phishing

Dieser Begriff bezieht sich auf betrügerische E-Mails, welche so aussehen, als ob sie von einer größeren Organisation (zB einer Bank) kommen würden und welche an einen großen Verteiler von Adressen verschickt werden. Diese E-Mails setzen stark auf „Sozialtechnik", sprich dem Empfänger wird suggeriert, dass er sensible Daten in ein ausführliches Formular eintragen muss, da ansonsten sein Bankkonto (im Falle einer Bank) gesperrt wird oder er im Falle einer Weigerung irgendwelche anderen Nachteile zu erwarten hat. Solche E-Mails enthalten in der Regel einen Link zu einer Website mit diesem Formular, welche vorgaukelt, sich auf dem Server besagter Organisation zu befinden. In Wahrheit jedoch wird der Nutzer an einen externen Server weitergeleitet, welcher im Design der tatsächlichen Organisationswebsite gehalten ist. Daher kann der Nutzer, welcher darauf herein-gefallen ist, keinen Unterschied feststellen und füllt die leeren Eingabefelder wie aufgefordert mit vertraulichen Informationen, Kontonummern, Passwörtern für das InternetBanking, PIN-Code für die Bezahlung usw aus. Die auf diesem Wege erhaltene Information kann von Betrügern leicht missbraucht werden.

13.1.2. SCHUTZ

Leider wird dem Begriff "Prävention" von PC-Nutzern nur wenig Bedeutung beigemessen. Nichtsdestotrotz ist es notwendig zu begreifen, dass das bloße Installieren von Antivirensoftware (auch wenn diese regelmäßig aktualisiert und korrekt konfiguriert ist) eine unzureichende Vorkehrungsmaßnahme ist. Regelmäßiges Updaten von zumindest all jenen Produkten, welche mit einem Computernetzwerk des Internets in Verbindung stehen, sind von zentraler Bedeutung. Kurz gesagt fällt darunter:

- das Zulassen regelmäßiger Updates des Betriebssystems Windows und seiner integralen Bestandteile (Start/Steuerkonsole/Automatische Updates)

- das regelmäßige Updaten anderer häufig verwendeter Produkte, welche vom Internet aus missbraucht werden könnten (Mozilla, Firefox, ICQ, DC++). Alternativ kann es ausreichen, regelmäßige Auto-Updates zu aktivieren. Klarerweise betrifft das auch zahlreiche „Plugins" von bestimmten Browsern (Java, Shockwave, Flash Player)

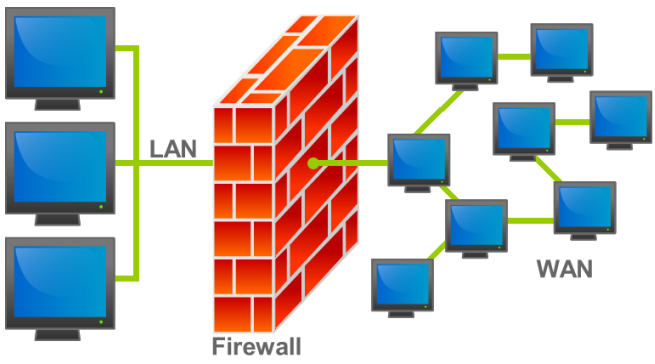
Antivirensoftware

Antivirensoftware sollte ein integraler Bestandteil jedes PCs sein. In den meisten Fällen ist ein großer Teil dieser Antivirensoftware auch ein Schutz vor Spyware. In diesem Zusammenhang ist es notwendig, Antivirensoftware und Anti-Spyware regelmäßig auto-

matisch updaten zu lassen. Angesichts der rasanten Geschwindigkeit, mit welcher sich Gefahren (hauptsächlich per E-Mail) verbreiten, ist ein Update jede halbe Stunde keine Utopie. Alles in allem lässt sich über Updates dieser Software sagen: Je öfter, desto besser. Ebenso wichtig ist es, stets Moduldurchläufe durchführen zu lassen, um Dateien zu finden, welche von Hackern oder Anwendungen manipuliert werden (öffnen/speichern/ausführen usw.) Solch ein Modul bezeichnet man gemeinhin als „Bürgerschild“ oder – etwas technischer – „On-Access Scanner“. Man muss sich auch bewusst sein, das dieses Modul der wichtigste Bestandteil einer Antivirensoftware ist. Obwohl derzeit in den meisten Fällen ein Modul angeboten wird, welches direkt heruntergeladene oder versendete Post inspiziert, mag es in Zukunft an Bedeutung verlieren, da eine mögliche Infektion ohnehin von dem On-Access-Sanner verhindert wird (mit ein paar Ausnahmen, welche Fehler in einem Programm ausnutzen). Obwohl ein Virus nicht identifiziert wird, bevor er im Posteingang landet, würde er in dem Moment erkannt werden, in dem ein Nutzer versucht, ihn zu öffnen, sprich den infizierten E-Mail-Anhang zu öffnen.

Firewall

Eine Firewall, zumeist eine persönliche Firewall, ist eine nützliche Ergänzung für all jene Nutzer, die auf das Internet zugreifen. Eine Firewall ist zu jenem Zeitpunkt nahezu unerlässlich, in dem sich ein Nutzer über seine IP-Adresse und seinen Computer mit dem Internet verbindet (ein Benutzer bekommt für gewöhnlich eine öffentliche IP-Adresse durch die Einwählverbindung oder eine Verbindung über das Internetkabel zugewiesen – Chello/UPC.) Ebenso wie Antivirensoftware muss eine Firewall sachgemäß verwendet werden. Daher sollte ein Nutzer in der Lage sein, zwischen einer legalen und einer illegalen Verbindung zu unterscheiden, sprich zu wissen, was erlaubt und was verboten gehört. Anderenfalls kann die Firewall ihren Zweck nicht erfüllen, unabhängig davon, ob man die in Windows integrierte Firewall oder eine andere, kostenpflichtige Firewall verwendet.



13.2. Gefährliches Netzwerk: Angriffe abwehren

Leider lauern in Netzwerken zahlreiche Gefahren. Es gibt tausende Applikationen, welche sich theoretisch und praktisch für Angriffe missbrauchen lassen. Vor manchen dieser Gefahren kann man sich relative leicht schützen, zB durch Beachten geeigneter Sicherheitsmaßnahmen. Dazu gehören starke Passwörter und das regelmäßige Installieren aktueller Software-Updates. Es gibt jedoch auch Gefahren, vor welchen man nicht gefeit ist und die äußerst schwer zu entdecken sind. Auf eben diese wird im weiteren Verlauf des Kapitels eingegangen.

Diese betreffen hauptsächlich Angriffe, welche über das TCP/IP-Protokoll ausgeführt werden. Dieses Protokoll ist über 30 Jahre alt und man fand heraus, dass nicht alles korrekt entworfen und angelegt wurde. Auf der anderen Seite muss man auch die Bedingungen anschauen, unter welchen das TCP/IP Protokoll designt und ins Leben gerufen wurde. Es ist evident, dass zum Zeitpunkt seiner Entwicklung unmöglich vorherzusehen war, dass das weltumspannende Computernetzwerk innerhalb weniger Jahrzehnte hunderte Millionen Nutzer haben würde. Mit größtmöglicher Vorsicht und durch das Einhalten geeigneter Sicherheitsmaßnahmen lassen sich jedoch auch vor nachfolgend beschriebenen Gefahren schützen.

Überwachung des Netzwerkverkehrs

Die Überwachung des Netzwerkverkehrs wird gemeinhin als “Sniffing” bezeichnet. Im Grunde handelt es sich hierbei um eine Analyse des Netzwerkverkehrs. Nachfolgend werden die Prinzipien und Bedingungen vorgestellt, nach bzw. auf deren Grundlage ein Angriff passieren kann. Zunächst muss man festhalten, dass diese Art von Angriffen für gewöhnlich in LAN-Netzwerken durchgeführt wird. Eine weitere Voraussetzung ist, dass das Netzwerk keine geschaltete Verbindung nutzt (diese Bedingung kann vernachlässigt werden, wie nachfolgend erklärt wird). An dieser Stelle wird erklärt, wie die Kommunikation in einem LAN-Netzwerk funktioniert. Alle Daten, unabhängig davon, woher sie kommen bzw. wohin sie gehen, durchqueren ein Übertragungsmedium (Kabel) in Form eines Clusters, welcher als „Frame“ bezeichnet wird. Jeder Frame ist für eine bestimmte MAC-Adresse konzipiert, sprich eine Adresse jeder Netzwerkkarte. In den meisten Karten ist die Adresse unveränderlich und vom Hersteller zugewiesen. Darüber hinaus erhält jede Karte im gleichen Netzwerksegment alle Frames, welche das Übertragungsmedium durchqueren und findet heraus, ob diese für sie bestimmt sind. Falls ja, gibt die Karte den Frame an eine höhere Schicht zur Verarbeitung weiter, ansonsten wird er ignoriert. Gleichwohl kann sich eine einmalige Situation ergeben, in welcher ein Frame für alle MAC-Adressen gültig ist (Rundfunk).

Gehen wir nun davon aus, dass Sie von einem Computer mit Verbindung zu einem LAN-Netzwerk aus mit einem lokalen Mail-Server verbunden sind. Sie geben dort Ihren Benutzernamen und Ihr Passwort ein und laden Ihre Post ohne Probleme herunter. Dabei bemerken Sie jedoch nicht, dass der Kollege neben Ihnen oder jemand am anderen Ende der Welt vielleicht ihr Passwort oder ihre Nachrichten liest. Dies ist jedoch durchaus möglich. Für den Kollegen ist dies ganz einfach: Es gibt schließlich Programme, welche

selbst dann das Auslesen von Frames erlauben, wen diese für eine andere MAC-Adresse bestimmt sind. Solche Programme nennt man Sniffing-Tools. Man muss das Programm nur installieren und starten – der Rest wird vom Programm erledigt. Einen fähigen Programmierer stellt das Schreiben eines solchen Programms vor keine große Herausforderung. Das Programm versetzt die Netzwerkkarte in einen sogenannten "Promiskuitätsmodus", welcher es ihm ermöglicht, den gesamten Verkehr in einem bestimmten Netzwerksegment zu überwachen und zu analysieren. Es gibt einfache Sniffing-Tools, zB UNIX TCPDUMP, oder äußerst fortgeschrittene wie Hunt. Effektiv vor Hunt schützen kann man sich nur mit entsprechender Verschlüsselung. Solange der Verkehr verschlüsselt ist, sind die Daten für einen Hacker nutzlos, da er sie nicht lesen kann. Die vorteilhaftesten Alternativen sind SSH und SSL. Daher ist es günstig, aktuelle Dienste zu ersetzen, welche Textpasswörter verwenden. Nichtsdestotrotz ist es notwendig zu wissen, wer den Verkehr überwacht. Handelt es sich um jemanden aus dem lokalen Netzwerk, lassen sich dagegen schwer Schutzmaßnahmen ergreifen. Es gibt jedoch Programme, welche in der Lage sind, Netzwerkkarten zu finden, welche sich aktuell im Promiskuitätsmodus befinden.

Denkbar ist aber auch, dass ein Sniffing-Tool auf einem Computer innerhalb eines LAN-Netzwerks durch einen Eindringling von außen installiert wird. Wenn es ihm gelingt, so ein Netzwerk zu hacken, wird er das höchstwahrscheinlich machen, da es keinen einfacheren Weg gibt, an Informationen über zahlreiche Benutzernamen und Passwörter heranzukommen. Der wirkungsvollste Schutz ist, das Netzwerk abzusichern, sodass niemand es hacken und ein Sniffing-Tool installieren kann.

DNS

DNS (Domain Name Service) bezieht sich auf einen der wichtigsten Dienste, den man in einem Netzwerk finden kann. Er stellt sicher, dass Domainnamen in IP-Adressen überführt werden und umgekehrt. Darüber hinaus macht dieser Service täglich das Leben von Millionen Benutzern leichter. Jedes Objekt (Server, Router), welches Teil eines IP-basierten Netzwerks ist (Intranet, Internet), besitzt eine exklusive Identifikation. Diese Identifikation wird durch die IP-Adresse repräsentiert. Hierbei handelt es sich um eine Nummer mit folgendem Format: xxx.xxx.xxx.xxx, zB 192.168.1.1. Diese Nummer ist exklusiv und kann nicht von zwei Objekten geteilt werden, welche gleichzeitig mit dem Netzwerk verbunden sind. Diese eindeutige Kennung hat jedoch eine namentliche Alternative, zB http://www.firma.cz, und die Um- und Rückwandlung zwischen diesen zwei Identifikatoren wird durch DNS sichergestellt. Man kann festhalten, dass sich Adressen wie http://www.pcworld.cz oder http://www.google.com leichter merken lassen als ein "bizarrer Zahlenmix". Im weiteren Verlauf dieses Kapitels wird die DNS-Funktionsweise im Detail beschreiben. Zu diesem Zweck wird in den folgenden Absätzen auf den Missbrauch von DNS eingegangen.

DNS-Manipulation

Eine DNS-Manipulation ist eine ziemlich gefährliche Aktivität. Um sie auszuführen, ist es notwendig, ein paar Anforderungen zu erfüllen. Unabdingbar ist in diesem Zusammenhang, dass der Hacker den Netzwerkverkehr überwacht (siehe oben). Unter der Voraussetzung, dass er Zugriff auf das LAN hat, kann er ein Programm auf seinem Computer starten, welches alle DNS-Anfragen abfängt und versucht, diese im Sinne des Hackers zu beantworten. Folglich ist das Hauptziel, die DNS-Antworten zu unterwandern und den Gast auf ein anderes System umzuleiten. So kann ein Hacker zB einen Netzwerk-Server haben, bei welchem es sich um eine perfekte Kopie eines E-Mail-Webservers handelt, welcher auf den ersten Blick identisch zum Original ist. Der Hauptzweck dieser Handlung besteht darin, alle Nutzer eines Netzwerks direkt auf diesen Server umzuleiten. Gibt ein Nutzer http://www.webmail.something mit der tatsächlichen Adresse 192.168.1.1 in seinen Browser ein (diese Adresse ist eigentlich nicht verfügbar, da sie sich auf die Adresse eines privaten Adressblocks bezieht, welcher für Netzwerke verwendet wird, die nicht direkt mit dem Internet verbunden sind; sie wird nur zu Illustrationszwecken verwendet), versucht das Hackerprogramm, die DNS-Antwort zu unterwandern und antwortet dem Client damit, dass der Server http://www.webmail.something eine 192.1691.100-Adresse hat. Im Fall, dass diese Antwort vor der Antwort des tatsächlichen und originalen DNS-Servers im Nutzersystem einlangt, gewinnt der Hacker. Wie lassen sich solche Angriffe vermeiden? Das erste und grundlegende Prinzip ist es, den Hacker am Eindringen in das Netzwerk zu hindern. Gelingt im das nicht, kann er nichts machen. Wenn ein autorisierter Nutzer sich jedoch auf diese Weise verhält, ist eine Suchmaschine für Sniffing-Tools eine umfassende Lösung. Eine weitere Option ist, einen DNS-Server zu verwenden, welcher seine Antworten signiert.

Andere Möglichkeit des DNS-Missbrauchs

Es gibt eine weitere Angriffsart, welche einen DNS-Server ausnutzt. Diese Variante ist jedoch schon ziemlich alt und nur bei älteren DNS-Versionen von BIND-Servern (UNIX DNS-Servern) effektiv, man stößt jedoch immer noch manchmal darauf. Diese Angriffsart wird Cache Poisoning genannt und setzt auf die Unterwanderung einzelner autorisierter DNS-Antworten. Um diesen Trick ausführen zu können, benötigt der Hacker nicht einmal Zugriff auf das Netzwerk. Die Lösung, um ihn auszusperren, ist jedoch relativ simpel: ein Software-Update.

Routing und Weiterleitung

Routing bezieht sich auf die Eigenschaften des IP-Protokolls, welches managt, auf welchem Weg Pakete durch das Netzwerk geleitet werden – unter der Bedingung, dass die Systeme, in die sich der Hacker unterwegs einklinkt, Routing zulassen. Mittels Routing können Pakete leicht gefälscht werden, wodurch man unautorisierte Informationen erhalten kann. Die optimale Lösung bestünde darin, Routing auszuschalten (aber weder Windows noch Linux stellen diesen Service in der Regel zur Verfügung). Nichtsdestotrotz sollte, so lange man Routing benötigt, die Zugriffssteuerungsliste (ZSL) in allen Routern

korrekt eingestellt sein.

Andererseits ermöglicht es Routing einem Hacker, auf das interne Netzwerk zuzugreifen, welches durch ein System verbunden sein kann und nicht für das Internet, das es umgibt, sichtbar sein muss. Dies ist unter der Bedingung möglich, dass die Weiterleitungsregeln nicht adäquat entworfen sind. Deshalb sollte man die Regeln stets überprüfen und testen.

Session Hijacking

Das "Entführen" einer Kommunikationssitzung bezeichnet eine fortgeschrittenere Angriffsmethode: Der Hacker benötigt ein fundiertes Wissen über Netzwerkkommunikation, was diesen Angriff besonders gefährlich macht. Wie zuvor muss der Hacker den Netzwerkverkehr überwachen. Ein ausgeklügeltes Gerät zum Hijacking von Sitzungen nennt man Hunt. Der Hacker startet das Hunt-Programm auf seinem Computer und beobachtet die Kommunikation zwischen zwei Systemen. Nachdem er den idealen Moment für das Hijacking der Sitzung abgewartet hat, versucht er, die Kontrolle darüber zu erlangen. Ist er erfolgreich, bekommt er die volle Kontrolle über die Sitzung und das Zielsystem findet nicht heraus, dass es mit einem fremden Subjekt kommuniziert. Alles in allem ist der Hacker nun in der Lage, all das zu machen, was der ursprüngliche Sitzungsinhaber machen kann. Adäquat schützen kann man sich davor nur durch geeignete Verschlüsselung und erhöhte Netzwerksicherheit.

Man In The Middle

Unter diesen Attacken versteht man weniger ausgereifte Methoden, welche leicht verhindert werden können. Im Wesentlichen hängt ihr Erfolg von der Naivität und Unreflektiertheit der Nutzer ab. Solche Angriffe können auch über verschlüsselte Protokollen wie SSH oder SSL ausgeführt werden. Kurz gesagt handelt es sich um keinen Protokollfehler, sondern um einen Missbrauch des Nutzervertrauens.

SSH

Wie schon der Titel dieses Angriffs andeutet, besteht ihr Grundprinzip darin, ein Kommunikations-medium bereitzustellen. Der Hacker muss die Sitzung eines Clients kapern, eine Verbindung zum tatsächlichen Zielserver herstellen und vorgeben, das Ende-zu-Ende-System zu sein. Dadurch fällt einem Client nicht auf, dass er über ein Drittsystem kommuniziert. Es gibt jedoch ein kniffliges Detail: Fakt ist, dass jeder SSH-Server seinen eigenen Identifikationsschlüssel hat. Da der Hacker in der Regel diesen Schlüssel nicht besitzt, muss er die Sicherheitsvorkehrungen umgehen. In den meisten Fällen erstellt er hierzu seinen eigenen Schlüssel. Dies führt jedoch dazu, dass einem Client, der sich mit dem System des Hackers verbindet, eine Warnmeldung angezeigt wird, die ihn auf die Änderung des Schlüssels aufmerksam macht. Zusammenfassend kann man sagen, dass der Erfolg dieser Attacke von der Gleichgültigkeit und der Sorglosigkeit des Nutzers ab-

hängt. Nur dann, wenn ein Nutzer alle Warnungen ignoriert und ohne jede Skepsis weitermacht, wird einem Hacker die Möglichkeit gewährt, den Verkehr zu verfolgen. Sobald einem eine derartige Ungereimtheit auffällt, sollte man den Server-Administrator kontaktieren.

SSL

Der Angriff über ein Medium kann auch auf das SSL-Protokoll angewendet werden. Das Prinzip ist dasselbe. Allerdings ist diese Angriffsart sogar noch gefährlicher, da das SSL-Protokoll innerhalb der Website verwendet wird. Hiermit interagieren unerfahrene Nutzer, welche es von den grafischen Benutzeroberflächen moderner Betriebssysteme gewohnt sind, blind jeder angezeigten Nachricht zuzustimmen. Deshalb ist es sehr wichtig, SSL-Zertifikate daraufhin zu überprüfen, ob sie von einer offiziellen Zertifizierungsbehörde signiert wurde. Vorsichtig sollte man vor allem bei selbst signierten Zertifikaten sein.

Passwörter

Selbst dann, wenn ein Netzwerk vollständig gesichert ist, alle Nutzer sich an die Sicherheitsrichtlinien halten und alles sorgfältig überwacht, kontrolliert und analysiert wird, kann ein Netzwerk immer noch angegriffen werden. Die meisten Server bieten bestimmte Dienste an, ohne die das Internet nicht wunschgemäß funktionieren würde. Auf diese Dienste können verschiedene Nutzergruppen über ein Netzwerk zugreifen. Manche dieser Services können vielleicht sogar von jedem genutzt werden, andere jedoch nur von bestimmten Personen. Unabhängig davon, wem ein Dienst angeboten wird, kann ein Angriff in der Regel durch das Netzwerk ausgeführt wird. Hacker nutzen hierfür eine Reihe verschiedener Methoden und Tricks. Stellt sich jedoch heraus, dass ein Netzwerk vollkommen sicher ist und die meisten der gebräuchlichen Angriffe fehlschlagen, könnte ein Hacker dazu übergehen, sich die Passwörter vorzunehmen. Nun ist es zwar so, dass diese Angriffe relativ zeitaufwändig sind und nicht immer zum Erfolg führen. Nichtsdestotrotz handelt es sich hierbei um die letzte Möglichkeit, in das System einzudringen. Passwörter aller Protokolle, welche zu einer bestimmten Form der Authentifizierung dienen, können angegriffen werden, zB SMTP, POP, FTP, SSH usw. Dieser Angriff kann von dutzenden Geräten ausgeführt werden. Nachfolgend gibt es einen Überblick über die verschiedenen Methoden des Passwort-Ratens und wie man es verhindert.

Wörterbuchangriff

Ein Wörterbuchangriff ist die häufigste Methode, Passwörter zu raten. Er erfordert einen schnellen Prozessor, ein optimiertes Programm und ein langes Wörterverzeichnis. Mithilfe eines geeigneten Programms ist es möglich, fast alle Passwörter anzugreifen. Das Raten von Passwörtern bezieht sich jedoch nicht ausschließlich auf Attacken, die über das Netzwerk ausgeführt werden: Passwörter können auch lokal geraten werden, was von besonderen Erfordernissen und Umständen abhängt. Das Grundprinzip besteht

darin, ein Wort zu nehmen, dieses in geeigneter Form anzupassen (in Bezug auf den Algorithmus) und es mit dem originalen Passwort abzugleichen. Netzwerkdienste sind darauf ausgerichtet, das Passwort über das Netzwerk zu übermitteln (seine Form wird vom Protokoll vorgegeben). Ein Wörterbuchangriff wählt Wörter aus, die als Passwörter in Frage kommen, und speichert diese in einer Datei ab. In der Folge werden verschiedene Zeichenkombinationen ausprobiert. Einige Wörterbücher sind sehr umfangreich, weshalb es günstig ist, ein Passwort zu wählen, welches sich nicht in einem bestimmten Wörterbuch befindet.

Passworterstellung

Sie sollten versuchen, ein zufälliges Passwort zu generieren und sich dieses irgendwo aufzuschreiben. Anschließend machen wir einen Test, ob das Passwort sorgfältig ausgewählt wurde. Nun benötigen wire in Wörterbuch. Ich empfehle, dass Sie sich Wörterbücher von der Plattform http://www.phreak.org/html/wordlists.shtml herunterladen. Darin finden sie eine große Auswahl an Wörterbüchern aus verschiedenen Sprachen. Sie können Wörterbücher herunterladen, die nach Themen oder Sprachen sortiert sind. Ich empfehle Ihnen, sie alle herunterzuladen. Nun gilt es herauszufinden, ob das gewählte Passwort sich darin findet. UNIX zB nutzt grep mojeheslo slovnik. Nachfolgend erhalten Sie ein paar Hinweise, wie sich ein starkes Passwort erstellen lässt.

<u>Was ein Passwort enthalten sollte:</u>

- Kleinbuchstaben des englischen Alphabets a-z
- Großbuchstaben des englischen Alphabets A-Z
- Ziffern 0-9
- Satzzeichen wie *, #, @,] etc.
- Mindestens sechs Zeichen

<u>Was ein Passwort nicht enthalten sollte:</u>

- ein echtes Wort
- einen Namen
- eine Kombination aus Name (Wort) und Zahlen, zB honza52
- persönliche Informationen, Spitznamen, den Namen der Frau usw.
- Zahlen, die man mit jemandem oder etwas in Verbindung bringen könnte
- Geburtstage, Telefonnummern, Adressen usw.

Es ist notwendig festzuhalten, dass ein Passwort nicht zweimal verwendet werden sollte. Legt man zB mehrere E-Mail-Adressen an, sollte man für jede davon ein eigenes Passwort vergeben. Es könnte sein, dass ein Passwort herausgefunden wird (zB durch ein Sniffing-Tool). Dies würde einem Hacker die Möglichkeit geben, auf mehrere andere Systeme zuzugreifen.

14. Literatur

Habraken, Joseph W. Průvodce úplného začátečníka pro Počítačové sítě : není zapotřebí žádných předchozích zkušeností!. 1. vyd. Praha : Grada, 2006. ISBN 80-247-1422-1.

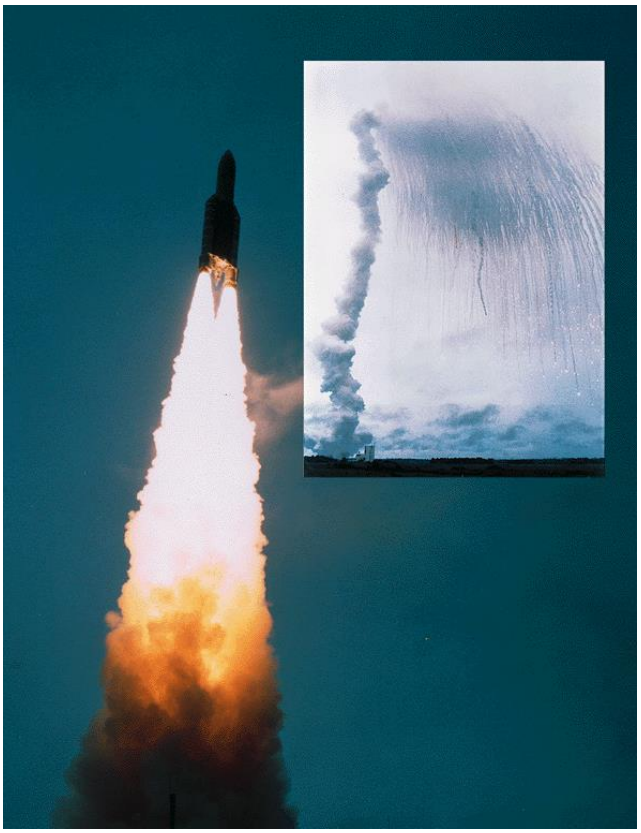
STALLINGS, William. Local and metropolitan area networks. 6th ed. Upper Saddle River: Prentice Hall, 2000. xvi, 478 s. ISBN 0-13-012939-0.

PUŽMANOVÁ, Rita. Moderní komunikační sítě od A do Z : [technologie pro datovou, hlasovou i multimediální komunikaci]. 2. aktualiz. vyd. Brno: Computer Press, 2006. 430 s. ISBN 8025112780.

THOMAS, Robert M. Lokální počítačové sítě. Vyd. 1. Praha: Computer Press, 1996. 277 s. ISBN 80-85896-45-1.

SOFTWAREENTWICKLUNG UND MODELLIERUNG

1. Einführung in die Softwareentwicklung



Softwarefehler können entweder einen Anwendungsabsturz auf einem Telefon oder ein wirklich großes Problem verursachen. Einige der negativen Folgen fehlerhafter Software werden in Richta's Vortrag (2011) aufgezeigt:

Absturz der Ariane-5-Rakete: Aufgrund eines Fehlers erteilte der Hauptrechner den Befehl, die Düsen der Feststoffraketenverstärkern und Motordüsen abzulenken. Dadurch wurde der Raketenkurs dramatisch verändert. Zusätzlich zu den aerodynamischen Kräften brach der obere Teil der Rakete ab. Anschließend wurde das Selbstzerstörungssystem der Rakete aktiviert und die Rakete in eine Wolke aus brennenden Scherben verwandelt. Gesamtsumme: 100 Mio. US $ (einschließlich Cluster-Satelliten, die Teil der Rakete waren, 500 Millionen Dollar).

Der Nordost-Ausfall von 2003 in den USA: Er wurde durch einen nicht registrierten Ausfall des automatischen Fehlerdetektors im Kraftwerk am Niagara River verursacht. Der Blackout verbreitete sich allmählich über das Netzwerk. Der Stromausfall hat insgesamt 21 Kraftwerke stillgelegt. Etwa 50 Millionen Menschen waren betroffen; drei Menschen starben; der Schaden betrug 6 Milliarden US $

Die Kollision des Landemoduls auf dem Mars (1999): Es gab ein Kommunikationsproblem zwischen den Komponenten - der Schnittstellenbenutzer erwartete einen Wert in Kilometern, der vom Anbieter jedoch in Meilen angegeben wurde. Statt 140-150 Kilometer über der Oberfläche geplant, landete er 57 Kilometer über der Oberfläche. In einer solchen Höhe ist die Marsatmosphäre zu schwer für eine Sonde. Der Climate Orbiter brannte wahrscheinlich in der Höhe von 80 Kilometern. (Quelle: Technet.cz)

1.1. Gründe für die Entwicklung von Software Engineering

Zu Beginn der Computerentwicklung in den 1940er und 1950er Jahren war die Computerrechenleistung sehr gering. Im Laufe der Zeit wurde diese Rechenleistung durch geometrische Reihen nach dem etablierten Moore'schen Gesetz entwickelt. Die Softwareentwicklung steckte noch in den Kinderschuhen. Es wurden keine Methoden oder Techniken entwickelt, welche die Entwicklung erleichtern würden. Als sich die Rechenleistung in den 1960er Jahren wieder deutlich weiterentwickelte, kam es zu einer so genannten Softwarekrise.

Die Besonderheiten der Softwarekrise waren unhaltbare Verzögerungen und Preiserhöhungen bei Projekten, schlechte Programmqualität, Wartungs- und Innovationsschwierigkeiten, geringe Effizienz der Programmierer, Entwicklungsineffektivität, schlechte Ergebnisse etc.

Die **Hauptursachen für die Krise** sind wie folgt *(Wikipedia)*:

- Schlechte Kommunikation zwischen den an der Softwareentwicklung beteiligten Personen und zwischen Entwicklern und Kunden.
- Falscher Ansatz. Die jeweilige Software wurde von einem Entwickler entwickelt und genehmigt und ein unzufriedener Kunde wurde als eine Person bezeichnet, die die Angelegenheit nicht versteht.
- Schlechte Planung des Projekts. Die Entwickler hofften, die Frist irgendwie einzuhalten.
- Unrealistische Schätzungen der Entwicklungsdauer, der Kosten und des Umfangs.
- Geringe Arbeitseffizienz. Die Programmierer achteten auf unwichtige Dinge und ignorierten das Wesentliche.
- Unkenntnis der Grundregeln. Z.B. das Gesetz von Brook aus dem Jahr 1975: "Das Hinzufügen einer Forschungskapazität zu einem verzögerten Projekt wird die Verzögerung nur noch verstärken."
- Unterschätzung von Bedrohungen und Risiken. Die Bedrohungen wurden nicht berücksichtigt und verursachten schließlich große Probleme.
- Schlechte Technologieanwendung. Falsche Vorstellung, dass alle Probleme nach der Einführung einer neuen Technologie gelöst werden.

Als Reaktion auf die oben genannten Themen fand 1969 eine Konferenz statt, die Grundlagen des Software-Engineerings vorstellte.

Damals war es wie folgt definiert:

"Software-Engineering ist eine Disziplin, die sich mit der Festlegung und Anwendung von Schlüsselingenieurprinzipien in der Softwareentwicklung beschäftigt, um eine wirtschaftliche Softwareentwicklung zu erreichen, die zuverlässig ist und effizient auf verfügbaren Computergeräten arbeitet."

Software Engineering kann als eine Ingenieurdisziplin bezeichnet werden, die sich mit aktuellen Problemen der Entwicklung komplexer Softwaresysteme beschäftigt. Diese Disziplin beinhaltet mehrere komplexe Denkprozesse. Diese Ingenieurdisziplin verfolgt pragmatische Ansätze. Diese Ansätze beinhalten die Kenntnis der speziellen Anforderungen an das Softwareprodukt, seine Analyse, sein Design, seine Implementierung, seinen Test und seine Installation beim Endanwender. Gleichzeitig beinhalten sie einen betriebswirtschaftlichen Ansatz für das Projektmanagement, der den effektiven Einsatz einzelner Techniken ermöglicht und deren Hauptziel es ist, ein qualitativ hochwertiges Softwareprodukt effektiv zu entwickeln.

1.2. Psychologische Exkursionen

Die Kognitionspsychologie erklärte verschiedene Wege der Problemlösung zwischen Experten und Novizen. Experten sind in der Lage, Teillösungen ähnlicher Situationen in Erinnerung zu rufen und das Vorwissen zur Identifizierung und Definition des Problems anzuwenden (Eysenck und Keane, 2008). Experten befassen sich mit dem Problem, indem sie den Kern der Sache und alle äußeren Umstände untersuchen. Auf der Grundlage dieser Informationen versuchen sie, eine Lösung zu finden. Anfänger hingegen versuchen, eine Lösung zu erraten und anschließend zu beurteilen, ob es möglich ist, den Lösungsvorschlag aus relevanten Fakten zu erarbeiten. Dabei nutzen sie die Rückwärtsshift-Strategie (Nolen Hoeksema at al., 2012). Laut Plháková liegt der größte Unterschied zwischen Novizen und Experten im Umfang der Wissensorganisation (Plháková, 2003). Hochentwickelte selbstregulierende Fähigkeiten (in einem separaten Kapitel untersucht) umfassen Merkmale erfolgreicher Individuen (Experten). Diese Fähigkeiten ermöglichen eine effektive Nutzung von Wissen und Fähigkeiten (Helus & Pavelková, 1992). Tatsächlich ist das Wissen von Laien, Probleme zu verstehen, meist sehr oberflächlich. Auf der anderen Seite nähern sich Experten zunächst dem Kern des Problems und versuchen, es zu identifizieren und zu organisieren. Sie nähern sich der Lösung erst, wenn sie einen realisierbaren Plan haben. Darüber hinaus konzentrieren sich die Experten viel länger auf die Vorbereitung als Laien, finden aber die optimale Lösung des Problems viel schneller (Plháková, 2003). Říčan (2016) sagt, zu Experten: "Sie verfügen nicht nur über mehr Wissen (quantitativer Aspekt), sondern unterscheiden sich auch qualita-

tiv, d.h. im Hinblick auf ein besser entwickeltes strategisches Verhalten bei der Bewältigung von Problemen (sie unterscheiden sich dadurch in ihren Lernprozessen). Experten haben besser organisiertes fachspezifisches Wissen, so dass sie neue Informationen aus ihrem Fachgebiet schneller erhalten können, da das vorhandene Vorwissen als Integrationsstruktur für neue eingehende Informationen über das assoziative Gedächtnis mit nur minimalem kognitiven Aufwand fungiert."

Im Allgemeinen können wir sagen, dass Software-Engineering Problemlösungsstrategien integriert, zu denen laut Kognitionspsychologie nur Experten in der Lage sind.

2. Lebenszyklus von Informationssystemen

Wikipedia vergleicht den Lebenszyklus treffend mit einer Brücke. Eine Brücke ist eigentlich ein Mittel, um menschliche Aktivitäten zu erleichtern. Dementsprechend sollten Softwaresysteme den gleichen Zweck erfüllen. Wie Brücken muss auch Software entworfen, genutzt, gewartet und schließlich außer Betrieb genommen werden. In der Bautechnik gibt es einen Lebenszyklus bestimmter Gebäude, der dem von Software sehr ähnlich ist.

Es gibt viele Klassifizierungen von Software-Lebenszyklen. Wir haben uns für den Systems Development Life Cycle (SDLC) des Justizministeriums der USA aus dem Jahr 2003 entschieden.

- Initiierung - erstens ist es notwendig, dass das Objekt mit genügend Geld (Kunde) den Bedarf an einer durch ein Softwaresystem implementierten Verbesserung verspürt. Nennen wir es "Absicht".

- Systemkonzeptentwicklung - diese Absicht wird auf Machbarkeit und Eignung überprüft. Der Systemumfang, auf dessen Basis die vom Kunden zu genehmigenden Basiskosten berechnet werden, ist lose definiert.

- Planungsphase - das Konzept wird so entwickelt, dass es beschreibt, wie das Unternehmen nach der Installation des genehmigten Systems funktionieren soll. Darüber hinaus werden spezifische Projektpläne erstellt und genehmigt.

- Phase der Anforderungsanalyse - alle Anforderungen sind detailliert beschrieben, was eine Weiterentwicklung des Systems ermöglicht. Diese Anforderungen betreffen in der Regel Daten, Rechenleistung, Sicherheitssystem, Wartungsanforderungen etc.

- Designphase - In dieser Phase werden die physischen Merkmale des Systems entworfen. Die Hauptteile des Systems, seine Ein- und Ausgänge werden definiert. Alles, was eine Eingabe oder die Zustimmung des Benutzers erfordert, muss vom Benutzer aufgezeichnet und überprüft werden.

- Entwicklungsphase - alle in den vorangegangenen Phasen festgelegten Spezifikationen werden in die entwickelte Software übernommen. Sie werden anschließend getestet.
- Integrations- und Testphase - einzelne Systemkomponenten werden integriert und systematisch getestet.

- **Implementierungsphase** - das System wird beim Kunden installiert und in Betrieb genommen. Der Kunde muss das System testen und freigeben. Die Phase endet mit der Inbetriebnahme des Produkts.

- **Betriebs- und Wartungsphase** - Der Betrieb des Systems sollte angemessen überwacht werden. Bei Bedarf werden notwendige Systemmodifikationen vorgenommen. Dieser Prozess geht weiter, bis das System effektiv an die Anforderungen des Kunden angepasst werden kann. Ist der Prozess nicht weiterführbar oder zu teuer, beginnt eine Phase der neuen Softwareplanung.

- **Dispositionsphase** - Außerbetriebnahme des Systems. Besonderes Augenmerk wird auf die genaue Speicherung der vom System verarbeiteten Daten gelegt, damit die Informationen im Falle eines zukünftigen Zugriffs effektiv in ein anderes System übertragen oder gemäß den offiziellen Vorschriften und Richtlinien der Aktenverwaltung archiviert werden können.

Der gesamte Software-Lebenszyklus wird im folgenden Text beschrieben. Einzelne Methoden der Softwareentwicklung werden gezeigt, um sich eng auf die RUP-Methode (Rational Unified Process) zu konzentrieren. Diese Methode umfasst Phasen von der Initiierung bis zur Implementierung. Die RUP-Methode arbeitet viel mit UML, daher werden sechs verschiedene Arten von UML-Diagrammen angezeigt. Geschäftsprozessmodell und Notation, die zu Beginn der Entwicklung neuer Software nützlich sein können, werden diskutiert. Softwaretests und deren Wartung mittels ITIL-Methode werden in einem separaten Kapitel beschrieben.

3. Methoden der Softwareentwicklung

Im Allgemeinen beziehen sie sich auf Verfahren, Regeln und Geräte, die zur Softwareentwicklung führen. Manchmal werden sie auch als Softwareprozess bezeichnet. Einzelne Geräte, die für die Softwareentwicklung konzipiert sind, werden als Framework bezeichnet. Ziel dieser Verfahren ist eine effektive Entwicklung und Wartung der Software.

Durch die Einhaltung dieser Verfahren reduzieren wir das Risiko unerwarteter Probleme und machen den Arbeitsplan einer bestimmten Software klarer. Wenn einzelne Entwickler die gleichen Verfahren befolgen, fördern sie damit die Zusammenarbeit durch die zuvor formulierten Regeln der Programmentwicklung.

Im Laufe der Zeit wurden viele Methoden der Softwareentwicklung entwickelt. Dennoch gibt es bisher keinen detaillierten und klar definierten Rahmen einer Softwareentwicklungsmethode, der als referentiell angesehen werden könnte. Zu diesen Methoden gehören:

3.1. Wasserfallmodell

Das Wasserfallmodell ist ein sequentieller und linearer Prozess, der als sequentieller Fortschritt betrachtet wird, der durch die einzelnen Phasen der Konzeption fließt: Design, Test, Integration, Wartung und alternativ Übergang. Dieses Modell stammt aus den 1970er Jahren. Es kann als ein wesentliches Modell betrachtet werden, auf dem andere Modelle basieren.

Der Wechsel von einer Phase zur anderen sollte nur dann durchgeführt werden, wenn die vorhergehende Phase teilweise oder vollständig verifiziert ist. Leider ist es oft nicht möglich, dieses Modell im wirklichen Leben zu nutzen. So ändert der Kunde beispielsweise die Programmspezifikationen oft erst in der Phase der Programmentwicklung oder bis zu deren Anwendung. Die Hauptprobleme sind wie folgt:

- Eine Beurteilung der Endqualität des Produkts im Rahmen der Softwareentwicklung ist aufgrund von zuvor festgelegten Kriterien für Softwarespezifikationen nicht möglich.
- Das Endprodukt hängt von den anfänglichen Anforderungen an die endgültige Software ab.
- Der Zeitaufwand für die Softwareentwicklung ist zu hoch.

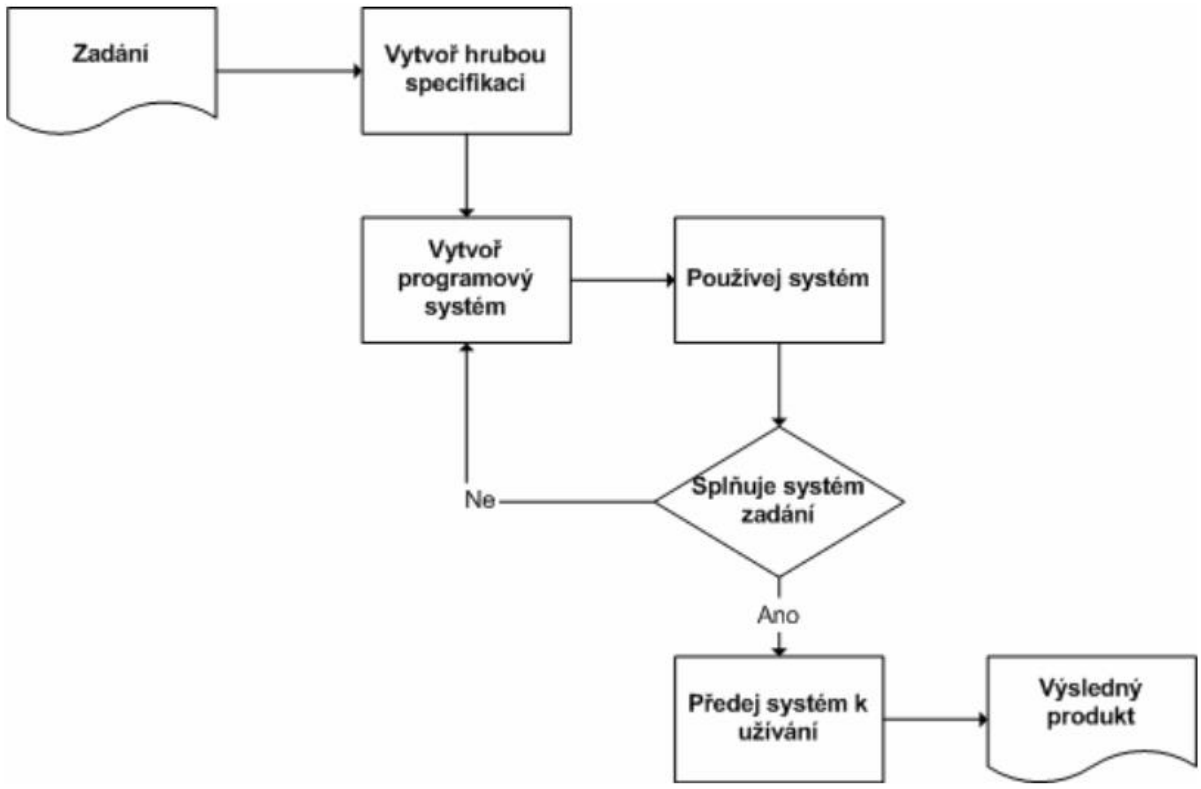
In Bezug auf dieses Modell würde der Versuch, Fehler zu beheben, zur Entwicklung anderer Modelle führen. Diese Prozesse werden von prof. Vondrák (2002) beschrieben:

3.2. Iterative und inkrementelle Entwicklung

Einzelne Schritte werden in wiederholten Zyklen durchgeführt "Iterationen". Jede Iteration nimmt einen kleineren Teil (Inkrement) der Funktionen in die Entwicklungssoftware auf, was zum Soll-Zustand führt. So wird das erforderliche Programm entwickelt und seine Funktionsfähigkeit verbessert. Damit hat der Kunde die Möglichkeit, die Entwicklungssoftware bereits in der frühen Entwicklungsphase auszuprobieren. Auf diese Weise kann er seine Anforderungen an die entwickelte Software besser spezifizieren. So werden Fehler, Mängel oder falsch gestellte Anforderungen an das Programm deutlich früher erkannt. Da der Kunde die Software von Anfang an sieht, kann er sich stärker an der endgültigen Form des Produkts beteiligen. Im Gegensatz zum Wasserfallmodell sind die Unterschiede zwischen den einzelnen Phasen sehr gering. Es ist vielmehr eine Parallele, in der die Iteration einen kleinen Wasserfall darstellt. Der RUP-Prozess, der in einem separaten Kapitel behandelt wird, fällt ebenfalls in diese Kategorie.

Explorative Programmierung

Prof. Vondrák (2002) betrachtet dieses Modell als abschreckendes Beispiel, das noch immer manchmal verwendet wird. Er nennt dieses Modell Exploratory Programming.



3.3. Agile Softwareentwicklung

Die meisten Softwareprozesse stammen aus Großunternehmen. Diese Prozesse erforderten die Einhaltung des geplanten Verfahrens und die obligatorische Erstellung vieler Dokumente. Das ist in großen Unternehmen angemessen, in denen viele Menschen an dem Projekt arbeiten oder in welchen ein hohes Maß an Zuverlässigkeit gefordert ist (z.B. Netzsteuerung). Bei kleinen Projekten war die Zeit, die damit verbracht wurde, alle Verfahren und sonstigen Formalitäten gehorsam zu befolgen, viel länger als die Zeit für Programmierung und Tests.

Als Reaktion darauf entstanden zu Beginn des neuen Jahrtausends agile Methoden oder Techniken. Die Grundprinzipien dieser Techniken wurden im Manifest für Agile Softwareentwicklung formuliert.

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge.
- Arbeitsprogramme sind wichtiger als eine umfassende Dokumentation.
- Die Zusammenarbeit mit dem Kunden ist wichtiger als die Vertragsverhandlung.
- Die Reaktion auf Veränderungen ist wichtiger als die Befolgung eines Plans.

Das Grundprinzip dieser Methoden ist es, den Entwicklern die Umgebung und Unterstützung zur Verfügung zu stellen, die sie für die Softwareentwicklung benötigen. Die Entwickler konzentrieren sich auf die Softwareentwicklung, ihr Design. Sie beschäftigen sich nicht zu sehr mit der Dokumentation. Diese Methoden fallen in iterative Methoden. Ihre hohe Priorität liegt in der aktiven Zusammenarbeit mit den Kunden. Der Arbeitsplan wird in der Regel innerhalb der nächsten Iteration erstellt. Aufgrund ihrer "anarchistischen" Natur eignen sich diese Techniken für kleinere Qualitätsentwicklerfirmen. Das zentrale Prinzip der Informationsvermittlung innerhalb eines Entwicklungsteams ist die persönliche Kommunikation.

Derzeit gibt es mehrere agile Methoden wie Extreme Programming und Scrum (Sklenář, 2007).

4. RUP

Der RUP-Prozess (Rational Unified Process) entstand in der zweiten Hälfte der 90er Jahre in Zusammenarbeit mit mehreren Unternehmen unter der Leitung der Rational Company. Dieses Unternehmen wurde 2003 in eine IBM-Abteilung umgewandelt. RUP ist nicht der einzige etablierte Prozess, es ist vielmehr ein iteratives Prozess-Framework, das an die Bedürfnisse von Entwicklerorganisationen und einem Software-Entwicklungsteam angepasst werden sollte. Diese Anpassung ist so implementiert, dass die Entwicklungsteams Prozesselemente auswählen, die ihren Bedürfnissen entsprechen. RUP stammt von UP (Unified Process). RUP gehören derzeit zu den weit verbreiteten Software-Prozessmodellen. Viele Tools unterstützen es.

Ein gemeinsames Wasserfallmodell definiert separate Rollen. Dabei kommt es vor, dass ein Analytiker die Vorstellung des Kunden vom Funktionieren des Systems akzeptiert. Diese Idee wird anschließend in ein Dokument umgesetzt, das dem Programmierer übergeben wird. Wenn es keine effektive Zusammenarbeit zwischen dem Analysten und dem Entwickler gibt, muss sich der Entwickler auf die im Dokument dargelegten Anforderungen konzentrieren. In diesem Fall kann die Interpretation der Anforderungen durch den Entwickler erheblich von der ursprünglichen Idee des Kunden abweichen.

Dieser Prozess basiert auf mehreren Prinzipien. Auch basiert er auf der Idee der software-iterativen Entwicklung. Dadurch sollte jedes Programm in Rekursionen (Iterationen) entwickelt werden. Die Iteration sollte durch einen ausführbaren Code erzeugt werden.

Žáček (2017) legt die Grundprinzipien von RUP fest:

- Der Versuch, Projektrisiken so schnell wie möglich und regelmäßig wie möglich zu minimieren.
- Um sicherzustellen, dass die Kunden einen Mehrwert erhalten.
- Fokussierung auf ausführbare Software.
- Veränderungen in frühen Phasen des Projekts vornehmen.
- Frühzeitiger Entwurf einer ausführbaren Architektur.
- Wiederverwendung vorhandener Komponenten.
- Enge Zusammenarbeit - alle sind ein Team.
- Die Projektqualität wird von all ihren Proportionen bestimmt, nicht nur von einem Teil (Testing).

Ursprüngliche Anforderungen an die Software sind für die Qualitätsentwicklung des Produktes unerlässlich. RUP erstellt ein **Anforderungsmanagementsystem**. Es verwaltet deren Erfassung, Dokumentation und überwacht Änderungen der Anforderungen. Anschließend wird dieses System von den Entwicklern in der Kommunikation mit den Kunden eingesetzt. Änderungen in der Softwareentwicklung werden auf die gleiche Weise verwaltet.

Viele Teile des Softwareprodukts sind sich ähnlich. Daher ist ihre Neuerfindung Zeitverschwendung. RUP führt die **komponentenbasierte Entwicklung** ein. Sobald wir über die notwendigen Komponenten verfügen, beginnt die Entwicklung des Produkts mit der Integration der entsprechenden Komponenten. Diese Integration kann mit Lego oder dem Aufbau eines Desktop-Computers verglichen werden.

Aufgrund der Komplexität der entwickelten Programme ist die **Softwarevisualisierung** für eine bessere Idee sehr wichtig. RUP arbeitet mit der UML-Sprache.

Die **Überprüfung der Softwarequalität** ist ein sehr wichtiger Aspekt für das Feedback an die Entwickler, um die Bedürfnisse der Kunden dank der funktionierenden Software zu erfüllen. An dieser Überprüfung sollten alle Produktentwickler beteiligt sein. RUP spezifiziert bestimmte Verfahren zur Beurteilung der Softwarequalität.

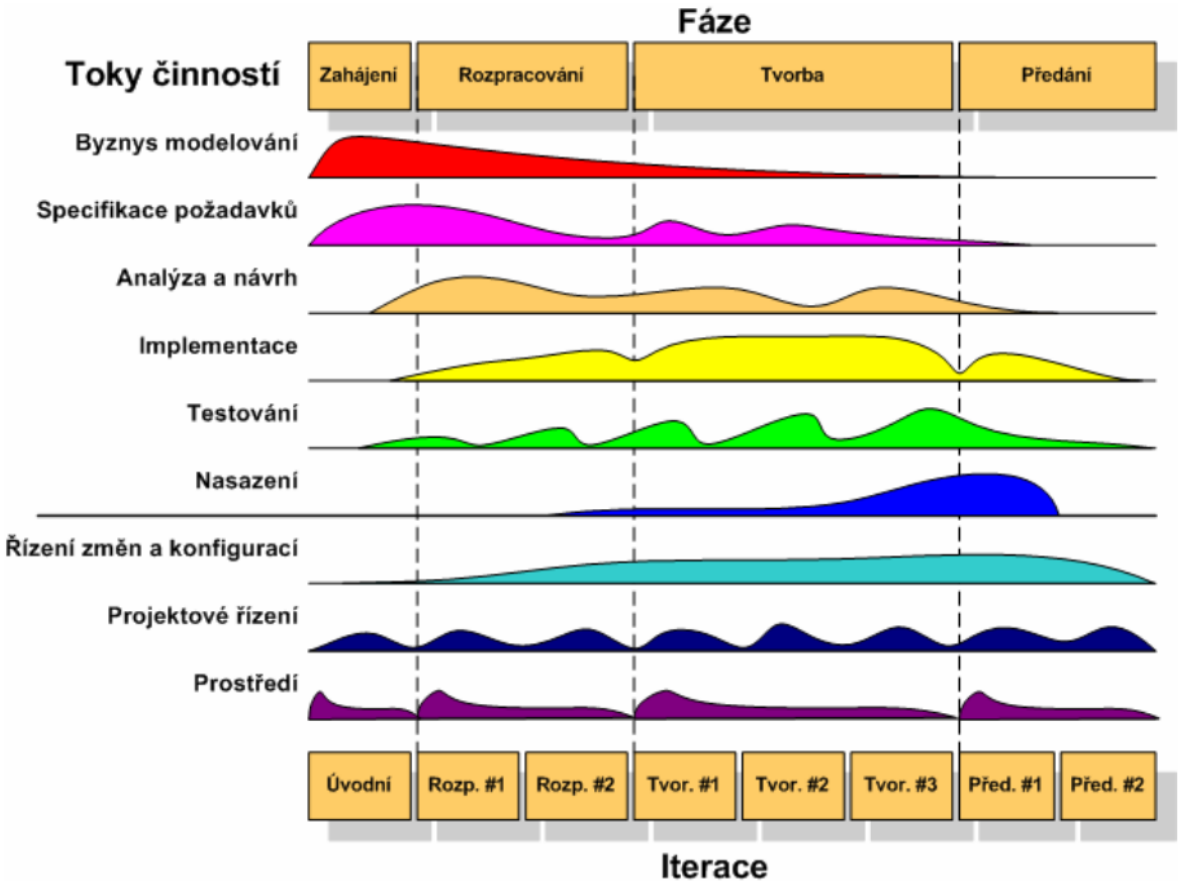
Die Softwareentwicklung von RUP gliedert sich in vier Phasen. RUP-Phasen können als einzelne Projektphasen und deren zeitliche Veränderungen bezeichnet werden. Jede Phase besteht oft aus mehreren Iterationen.

4.1. Schematisches Modell

Das schematische Modell besteht somit aus sechs Ingenieurdisziplinen und drei Hilfsdisziplinen. Zu den ingenieurwissenschaftlichen Disziplinen gehören:

- Geschäftsprozessmodellierung
- Anforderungsspezifikation
- Design und Analyse
- Implementierung
- Prüfung
- Bereitstellung
- Zu den Hilfsdisziplinen gehören:
- Konfigurations- und Änderungsmanagement
- Projektleitung
- Vorbereitung der Projektumgebung

Eine relative Häufigkeit von Aktivitäten einzelner Disziplinen im Zusammenhang mit einzelnen Phasen und Projektiterationen wurde von prof. Vondrák (2002) erarbeitet:



Eine relative Zeit- und Quellverteilung sollte wie folgt aussehen:

	Beginn	Ausarbeitung	Errichtung	Übergang
Quellen	Ca 5%	20%	65%	10%
Zeit	10%	30%	50%	10%

Žáček (2017) schlägt die Ergebnisse der einzelnen Phasen vor:

- Das Ergebnis von Inception ist das Verständnis der wichtigsten Themen, das Ziel des Projekts, die Bewertung der Risiken.
- Der Output von Elaboration ist eine ausführbare und verifizierte Architektur (=Funktionierender Teil der Anwendung).
- Der Output von Construction ist eine Beta-Release-Anwendung - eine relativ

stabile, wiederausführbare und nahezu vollständige Anwendung.

- Das Ergebnis von Transition ist ein Produkt, das für den endgültigen Einsatz bereit ist, einschließlich der gesamten Dokumentation und Hardware. Der wesentliche Unterschied besteht auch darin, dass jede Phase wie folgt charakterisiert ist:

  Die einzelnen Phasen umfassen eine unterschiedliche Anzahl von Mitarbeitern und Stellen. Es ist wichtig, dass die Mitarbeiter immer in einem oder mehreren Teams zusammenarbeiten. Bei Bedarf helfen und beraten sie sich gegenseitig.

4.2. Beginn der Tätigkeit

Zu Beginn ist es notwendig, die Projektgröße, den Bedarf, die Zeit- und Kosten der gesamten Projektdurchführung zu beurteilen. Žáček (2017) schlägt 5 Ziele dieser Phase vor:

- Den Prozess verstehen - die Vision vermitteln, die Systemgröße und -grenzen definieren; die Person verstehen, die das System will und wie sie davon profitieren könnte.
- Identifizierung der wichtigsten Systemfunktionalitäten - Identifizierung der kritischsten Use Cases.
- Vorschlag mindestens einer möglichen Lösung (Architektur), (d.h. ob es möglich ist, in der Entwicklung fortzufahren Hinweis: Autor)
- Über Kosten, Projektkonzeption und Risiken informiert zu sein.
- Definition/Modifikation des Prozesses; Werkzeugauswahl und -einstellung.

Definition und Auferlegung von Anforderungen

Im Rahmen der gestellten Anforderungen versuchen wir, maximale Informationen darüber zu erhalten, was die Benutzer von dem neuen System erwarten und welche wesentlichen Funktionen das System erfüllen soll. Zu den Techniken der Datenerhebung gehören:

- Dokumente und Aufzeichnungen
- Interviews mit zukünftigen Anwendern
- Fragebögen
- Beobachtung
- Aufzeichnung der Aktivitäten der Benutzer

Diese Anforderungen lassen sich unterteilen in:

- **Funktionale Anforderungen** Dies sind Anforderungen an die Systemfunktionalität aus dem System; alternativ sollten die Funktionsfähigkeit und Unfähigkeit des Systems definiert werden. Ein Anwendungsfalldiagramm ist ein effektives Werk-

zeug zur Veranschaulichung der funktionalen Anforderungen.

- **Nichtfunktionale Anforderungen** Dies sind Anforderungen an die Systemeigenschaften als Ganzes. Sie können sich auf Effizienz, Zuverlässigkeit, Sicherheit und Systemkapazität beziehen; alternativ können sie sich auf organisatorische Anforderungen wie die Einhaltung von Standards und die Einhaltung etablierter Verfahren beziehen. Alternativ können sie auch Anforderungen an die gültige Gesetzgebung beinhalten.

Ziele

Das Hauptziel ist es, eine Vision des Projekts zu vermitteln. Diese Vision sollte in Bezug auf den Benutzer definiert werden. Sie sollte jedoch gleichzeitig eine grundlegende technische Sicht auf die angewandten Technologie- oder Architekturkomponenten enthalten. In dieser Vision sollten vor allem die Anforderungen an die zukünftige Software festgelegt werden. Diese Phase findet nur einmal statt. Dennoch kann sie in komplexeren Fällen mehrfach wiederholt werden.

Gleichzeitig sollte in bestimmten Bereichen der Grad der Risiken bewertet werden; diese Risiken stellen sich wie folgt dar:

- Risiken technischer Bereiche (Technologien, Kompatibilität, Kommunikation mit anderen Systemen)
- Finanzielle Risiken - Entwicklungskosten und anschließende Wartung
- Zeitrisiken - je mehr Quellen vorhanden sind, desto kürzer ist wahrscheinlich die eigentliche Entwicklung.

Am Ende dieser Phase ist es notwendig zu beurteilen, ob die Entwicklung fortgesetzt werden kann. Je früher wir einer obskuren Projektentwicklung ein Ende setzen, desto geringer sind die Kosten der Projektentwicklung. Zu diesem Zweck wird der Lifecycle Objective Milestone (LOM) mit folgenden Bewertungskriterien verwendet:

- Interessierte Parteien stimmen über den Projektumfang, die Kosten und den Zeitplan überein.
- Vereinbarung, dass die richtigen Anforderungen an das System gestellt wurden und dass es ein gemeinsames Verständnis dieser Anforderungen gibt.
- Vereinbarung, dass die Kosten- und Terminschätzungen, Prioritäten und Risiken dem Entwicklungsprozess entsprechen.
- Alle Risiken wurden identifiziert und Minderungsstrategien festgelegt.

4.3. Ausarbeitung

Diese Phase beginnt mit der Bildung des Projekts. Ziel dieser Phase ist es, die Systemar-

chitektur zu etablieren, um in der nächsten Phase einen Großteil der gesunden Funktionalitäten zu produzieren. Diese Architektur basiert auf Erkenntnissen aus der vorherigen Phase. Wesentliche Fragen des Architekturdesigns der entwickelten Software sollen hier behandelt werden. Die Funktionalität des Systems muss nicht perfekt sein, dennoch sollte der Großteil der wesentlichen Funktionalitäten bereits in das System integriert sein. Das Forschungsteam sollte sich darüber im Klaren sein, dass sich das Architekturdesign in dieser Phase stabilisieren muss.

An dieser Stelle wird das entscheidende Thema analysiert und die Projektarchitektur erhält ihre Grundform. Mit dem Architekturvorschlag bemühen wir uns die Risiken im Zusammenhang mit der Softwareentwicklung zu minimieren. Diese Risiken betreffen im Wesentlichen:

- Zeit und Finanzen
- Genauigkeit der Architektur
- Prozesse und Technologien
- Ziel der Softwareentwicklung

Eine Reihe kleinerer Iterationen können den Grad der oben genannten Risiken an dieser Stelle mindern und helfen, das Hauptziel der Entwicklung zu überwachen. Wenn wir das System mit ähnlichen Technologien wie bisher aufbauen, gibt es offensichtlich ein geringeres Maß an potenziellen Risiken. Daher sind wir in der Lage, die Ziele dieser Phase innerhalb der Iteration zu erreichen. Andererseits, wenn wir mit der verwendeten Technologie nicht vertraut sind oder das Projekt komplexer ist (z.B. strengere Sicherheitsanforderungen), benötigen wir zwei oder mehr Iterationen. Für den Fall, dass in dieser Phase eine wesentlichere Änderung der Architektur stattfindet, sollte eine weitere Iteration hinzugefügt werden, die die notwendige Funktionalität und Stabilität der modifizierten Architektur überprüft. Hier gilt der Grundsatz "je später das Fundament der Konstruktion geändert wird, desto teurer wird der gesamte Umbau".

Um das Ziel der Softwareentwicklung zu überprüfen, sollte eine Beta-Version der Benutzeroberfläche erstellt werden, auf der die wichtigsten Systemfunktionalitäten getestet werden.

Am Ende dieser Phase muss die Systemarchitektur stabilisiert werden. Die ausführbare Architektur muss die Systemarchitektur überprüfen und dient gleichzeitig als Mittel zur Überprüfung kritischer Systemfunktionalitäten. Diese kritischen Funktionalitäten müssen auch modelliert werden, z.B. mittels UML-Sequenzdiagramm. Wir versuchen auch, potenzielle Probleme und Risiken zu identifizieren. Anschließend sollten wir ausgewählte Objekte zu Paketen gruppieren, die Sichtbarkeitsregeln und zukünftige Produktkonfiguration betreffen. Wir sollten auch eine Vorstellung davon haben, wie mit den Daten in der Datenbank umgegangen wird, weshalb wir die Komponenten regelmäßig integrieren sollten. Das bedeutet, dass die Reihenfolge und Art und Weise, in der die ausgewählten Komponenten integriert werden, festgelegt wird.

Schließlich stellt das Testen kritischer Szenarien einen sehr wichtigen Teil dieser Phase dar. Das Testen kritischer Szenarien kann unseren Fortschritt in der Erarbeitungsphase belegen. Darüber hinaus sollten kritische Szenarien unter Aspekten wie hohe Systembelastung, auf ausreichende Systemarchitekturleistung oder Zusammenarbeit mit externen Systemen getestet werden.

An dieser Stelle testen wir die Systemvorgaben mittels Lifecycle Architecture Milestone (LCA), während es sich bei den Bewertungskriterien um LCA handelt:

- Die Produktvision und die Anforderungen sind stabil.
- Die Architektur ist stabil.
- Die wichtigsten Ansätze und Verfahren werden getestet und verifiziert, so dass sie sich als anwendbar erwiesen haben.
- Tests und Evaluierungen von ausführbaren Prototypen haben wichtige Projektrisikoelemente gezeigt, die erfolgreich gelöst wurden.
- Die Iterationspläne für die folgende Phase werden erstellt, damit die Arbeiten fortgesetzt werden können.
- Die Iterationspläne werden durch glaubwürdige Schätzungen unterstützt.
- Alle Teilnehmer sind sich einig, dass die aktuelle Vision erfüllt werden kann, wenn der aktuelle Plan im Kontext der aktuellen Architektur ausgeführt wird.
- Istkosten im Vergleich zu Plankosten sind akzeptabel.
- Das Projekt kann abgebrochen oder anderweitig überdacht werden, wenn es den Meilenstein nicht erreicht.

4.4. Konstruktion

Das Hauptziel dieser Phase ist es, ein glaubwürdig funktionierendes Softwaresystem aufzubauen und gleichzeitig die Kosten zu minimieren.

Diese Phase konzentriert sich auf die Entwicklung von Komponenten und anderen Systemelementen. Diese Phase ist weiterhin dadurch gekennzeichnet, dass sie eine Verschlüsselung schafft, die sehr zeitaufwendig und menschlich ist (es erfordert eine Menge Programmierer und Tester). Der Rest der Funktionalitäten wird hier entworfen, d.h. die restlichen Hauptfunktionen (Kundenwünsche) und die meisten anderen. Wenn ein Eingriff in die Systemarchitektur behandelt werden soll, ist es wahrscheinlich, dass die vorherige Phase falsch abgeschlossen ist.

Wesentliche Voraussetzungen für den erfolgreichen Abschluss dieser Phase sind: Architekturkompaktheit, parallele Entwicklung einzelner Teams, Konfigurations- und Änderungsmanagement sowie automatisiertes Testen.

Größere Projekte können mehrere Iterationen durchführen (üblicherweise 2 - 4). Diese

Phase führt in der Regel die meisten Iterationen durch. Die Iterationsplanung entspricht der Anzahl und Art der noch nicht implementierten Funktionalitäten. Jede Iteration befasst sich mit einem separaten Projektsegment. Im Idealfall entsteht die Architektur, die weiterentwickelt, genutzt oder wiederverwendet wird, aus früheren Phasen. Das System wird integriert und getestet. Im Hinblick auf die effektive Organisation ist ein Team für die Architektur verantwortlich, während andere Teams parallel an der Entwicklung des Subsystems arbeiten. Diese "parallelen Teams" kommunizieren hauptsächlich mit dem Team, das für die Softwarearchitektur verantwortlich ist.

Dank der iterativen Entwicklung werden viele Dateien und deren Versionen entwickelt, die anschließend getestet und integriert werden. Aus diesem Grund muss ein Konfigurations- und Änderungsmanagement eingeführt werden. Dieses Management registriert individuelle Konfigurationen und Änderungen. Funktioniert ein solches System, können sich die Entwickler ganz auf die Weiterentwicklung konzentrieren und so ihre Effektivität steigern.
Am Ende dieser Phase muss eine Beta-Version dieses Programms entwickelt werden, die Hilfe bei der Anwendung, Installationsanweisungen, Benutzerhandbücher und Tutorials enthält. Auf diese Weise können zukünftige Anwender die Software ausprobieren und uns ein konstruktives Feedback geben.

Am Ende dieser Phase sollten wir uns mehrere Fragen gemäß dem IOC-Meilenstein (Initial Operational Capability) stellen.

- Die Beta-Version ist bereit für die Freigabe durch die ersten Benutzer (Tester).
- Die Benutzer sind vorbereitet und können unsere Beta-Version nutzen.
- Istkosten versus Plankosten sind akzeptabel.

4.5. Übergang

Diese Phase zielt auf den Übergang des Systems von der Entwicklung zur regelmäßigen Nutzung und die Behebung der wichtigsten Mängel ab (sie betreffen in der Regel die Leistung, Benutzerfreundlichkeit und Funktionalität). Zu diesen Phasenaktivitäten gehören:

- Schulung von Administratoren und Endanwendern
- Systemtests zur Überprüfung der Erfüllung der Erwartungen der Endanwender
- Erstellung von Marketingmaterialien
- Umgebung und Datenaufbereitung, z.B. Datenmigration vom alten zum neuen System
- Das Produkt wird in Bezug auf die zu Beginn des Projekts festgelegten qualitativen Rahmenbedingungen genauestens überwacht.
- Interne Projektbewertung, Lernen aus Fehlern und Problemen aus der Projektarbeit.

Englisches Wikipedia vom 24.6.18 weiters erwähnt: "Das System durchläuft auch eine Evaluierungsphase; jeder Entwickler, der keine sinnvolle Arbeit verrichtet, wird ersetzt oder entfernt.

Auch diese Phase sollte mit einem Meilenstein abgeschlossen werden. PRM (Product Release Milestone) Kriterien sind:

- Die Benutzer sind zufrieden.
- Die Endkosten im Vergleich zu den geplanten Kosten sind akzeptabel. Wenn die Kosten überstiegen wurden, was hätte man tun müssen, um das Problem zu umgehen?

5. Geschäftsprozessmodell und Notation

Ziel war es, ein grafisches Werkzeug zu entwickeln, das für alle Geschäftsanwender (von Analysten über Entwickler bis hin zu Geschäftsprozessverantwortlichen) verständlich ist, was eine effektivere Kommunikation zwischen Entwickler und Kunde ermöglicht. Geschäftsprozessmodell und Notation (nachfolgend BPMN genannt) besteht aus einer Reihe einfacher grafischer Elemente, aus denen ein Geschäftsprozessmodell modelliert werden kann. BPMN wird seit 2005 entwickelt. Die Version 2.0. ist im Jahr 2011 entstanden. Das Prozessablaufdiagramm basiert auf einem Flussdiagramm, das dem UML-Aktivitätsdiagramm sehr ähnlich ist, das im weiteren Text erläutert wird.

Flow-Objekte

Es ist eng mit dem Informationsfluss im Prozess verbunden. Flow-Objekte sind die wichtigsten grafischen Elemente. Sie können in drei Gruppen eingeteilt werden - Events, Activities und Gateways.

Events

Ereignisse werden durch einen Kreis gekennzeichnet, in dem ein Symbol angezeigt werden kann. Es markiert ein Ereignis, das den Prozessablauf direkt steuert. Ereignisse werden in Startereignisse unterteilt, die den Prozess auslösen und mit einem Kreis mit schmalem Rand - manchmal mit grüner Farbe - markiert sind.

Darüber hinaus gibt es Endereignisse, die das Ende des Prozesses oder das Ergebnis der Aktivität anzeigen. Diese sind mit einem fetten Rand oder einem fetten Symbol im Kreis markiert - manchmal mit roter Farbe. Einige Quellen erwähnen auch Intermediate Event, das mit einem doppelten Rand markiert ist.

Aktivität

Aktivitäten werden durch ein abgerundetes Rechteck markiert. Dieses Element beschreibt Arbeit oder Tätigkeit. Die Aktivität kann entweder atomar (d.h. Aufgabe) sein, was als eine einzige Einheit betrachtet wird und immer als Ganzes ausgeführt werden muss.

Alternativ kann es auch auf einen einzelnen Prozess heruntergebrochen werden, der als Subprozess bezeichnet wird. Diese Art von Aktivität wird in Prozessen eingesetzt, die wir auf einer bestimmten Ebene nicht offenbaren wollen. Teilprozessaktivitäten werden durch ein Pluszeichen in der unteren Zeile des Rechtecks mit abgerundeten Ecken gekennzeichnet.

Gateway

Gateways sind mit einer quadratischen oder diamantförmigen Form gekennzeichnet. Es bestimmt die Verzahnung und Zusammenführung von Prozessabläufen, z.B. bei Entscheidungen oder Parallelverarbeitung.

Objekte verbinden

Sie verbinden Flussobjekte oder Artefakte. Durch die Verbindung entsteht eine neue Struktur (Framework) des Geschäftsprozesses. Die verbindenden Objekte werden in drei Grundtypen unterteilt:

- Sequence Flow - ist mit einer durchgezogenen Linie und einer Pfeilspitze gekennzeichnet. Es zeigt die Reihenfolge, in der die einzelnen Flussdiagrammeinheiten ausgeführt werden sollen.

- Nachrichtenfluss - ist mit einer gestrichelten Linie und offener Pfeilspitze gekennzeichnet. Es zeigt den Nachrichtenfluss über die Unternehmensgrenzen des Prozesses hinweg an.

- Assoziation - ist mit einer gestrichelten Linie gekennzeichnet. Es ordnet Objekten zusätzlichen Informationen zu. Es wird auch für die Anzeige von Ein- und Ausgängen verwendet.

Schwimmstrecken

Sie zeigen die Teilnehmer des Prozesses an und beziehen sich auf die Organisation und Kategorisierung der Aktivitäten im Prozess. Sie bestehen aus zwei Arten.

- Pool - repräsentiert die Teilnehmer des Prozesses, oder alternativ trennt er verschiedene Organisationen. Es kann mehr Spuren enthalten. Ein Pool enthält einen separaten Prozess.

- Lane - befindet sich unter dem Pool. Es wird für die Organisation und Kategorisierung von Aktivitäten verwendet.

Artefakte

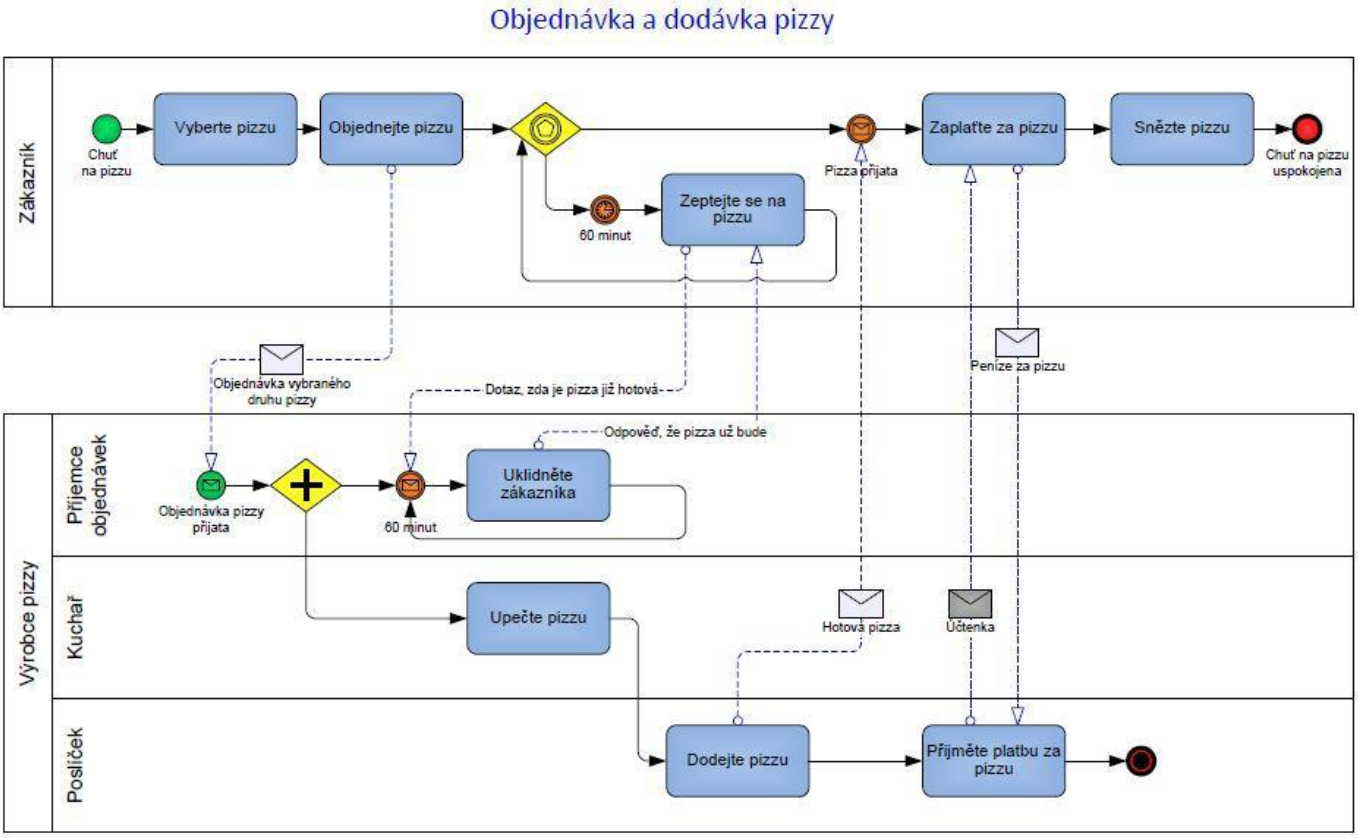
Sie bringen einige neue Informationen in den Prozess ein. Sie sind unabhängig von Bewegungen.

- Datenobjekt - ist mit einem Rechteck in der Biegeecke (ein Blatt Papier) markiert. Es zeigt die in einer Aktivität produzierten Daten an, oder es sagt uns, welche Daten für eine Aktivität benötigt werden.

- Gruppe - ist mit einem abgerundeten Rechteck und gestrichelten Linien markiert. Es wird verwendet, um verschiedene Objekte zu gruppieren.

- Annotation - vermittelt dem Leser einen klaren und verständlichen Eindruck.

Wir stellen ein Modell der Bestellung und Lieferung von Pizza als Beispiel zur Verfügung.



6. UML

Tatsächlich sind viele Softwaresysteme außergewöhnlich komplex. So liegt ihre Entwicklung mit strengen Anforderungen in den Händen und in der Phantasie der Entwickler. Wenn an einem Projekt mehrere Menschen beteiligt sind, muss das zukünftige Softwaresystem genau definiert - modelliert - werden. Es gibt viele Ansätze für eine solche Modellierung. Alle haben ihre Vor- und Nachteile. Um eine effektive Zusammenarbeit zu fördern, ist es wichtig, dass jeder das Modellierungssystem und seine Terminologie kennt, verwendet und versteht. Aus diesem Grund wurde die UML (Unified Modeling Language) entwickelt. Die Standard-UML wird von der Object Management Group (OMG) definiert.

UML bezieht sich auf eine einheitliche Sprache zum Zeichnen von Diagrammen. UML ermöglicht Spezifikationen (Struktur und Modell), Visualisierung (Flussdiagramme), Konstruktion (Softwareentwicklungsmethoden) und Dokumentation von Software-Systemartefakten. Es gibt eine große Anzahl von Diagrammen. Jedes dieser Diagramme wird für ein anderes Ziel und in verschiedenen Projektphasen verwendet. UML bietet insgesamt 13 Arten von Flussdiagrammen. In diesem Text werden die Diagramme in Strukturdiagramme und Klassendiagramme unterteilt.

Strukturdiagramme

Strukturdiagramme beschreiben die Projektstruktur. Sie werden oft verwendet, um die Architektur der entwickelten Software zu beschreiben. Wir werden uns mit mehreren ausgewählten Diagrammen dieser Kategorie befassen.

Klassendiagramm

Dieses Diagramm zeigt die Projektklasseneinteilungen, ihre Beziehungen, Operationen und Methoden oder Methoden der Methoden. Diese Klassen sind in einem rechteckigen, vertikal in 4 Teile gegliederten Rechteck geschrieben. Dieses Diagramm ist der Grundstein der objektorientierten Modellierung. Es wird auch für die Datenmodellierung verwendet. Darüber hinaus stellt das Diagramm eine statische Struktur des Systems dar und hängt teilweise von einer bestimmten Programmiersprache ab. Ziel dieser Aktivität ist es, die Art und Weise vorzuschlagen, wie das Produkt in der Implementierungsphase implementiert wird.

Beispiele für Klassendiagramme finden Sie unten.

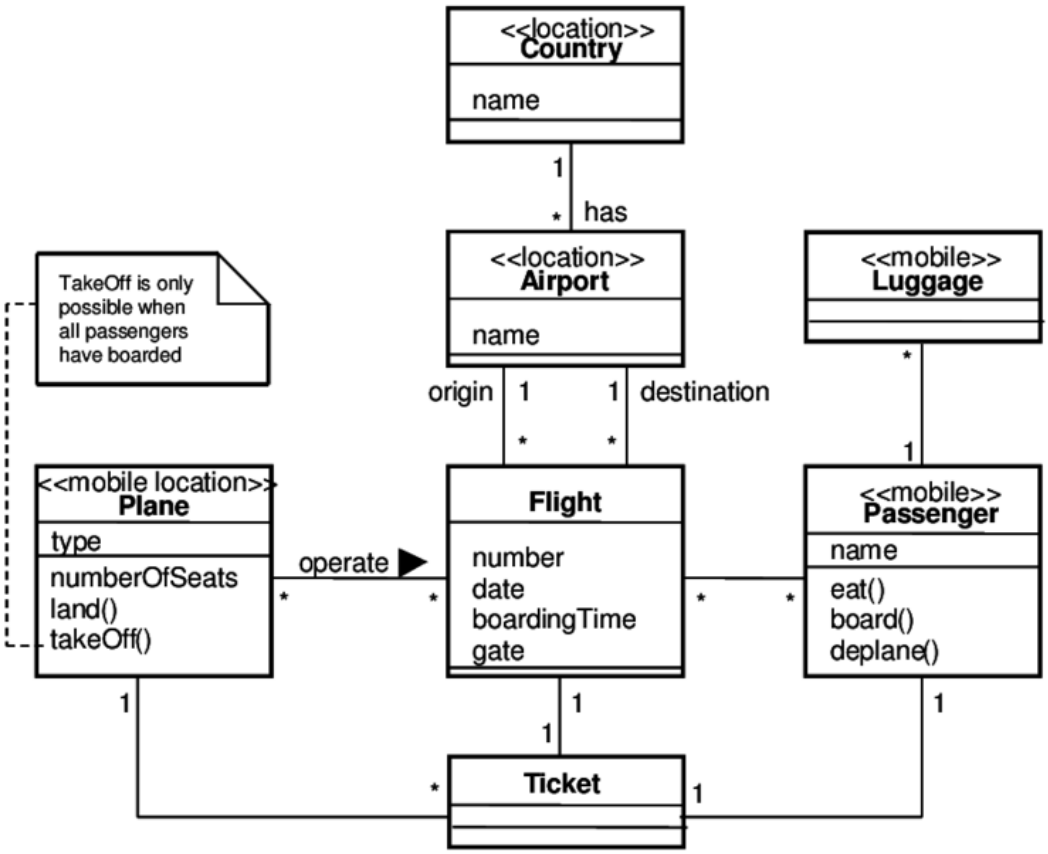


Abb.: Quelle - Andrade et al, Recent Trends in Algebraic Development Techniques—16th International Workshop, WADT 2002, Frauenchiemsee, Germany. Vol. 2755 of LNCS.

Komponentendiagramm

Dieses Diagramm arbeitet mit den im System verwendeten Komponenten und beschreibt, wie die Komponenten zu größeren Komponenten von Softwaresystemen verbunden werden. Es wird verwendet, um die Struktur verschiedener komplexer Systeme zu beschreiben. Das Diagramm wird hauptsächlich für den Entwurf von Software nach Komponenten verwendet. Es hat auch eine höhere Abstraktionsebene als Klassendiagramme. Eine oder mehrere Klassen sind innerhalb einer einzigen Komponente implementiert.

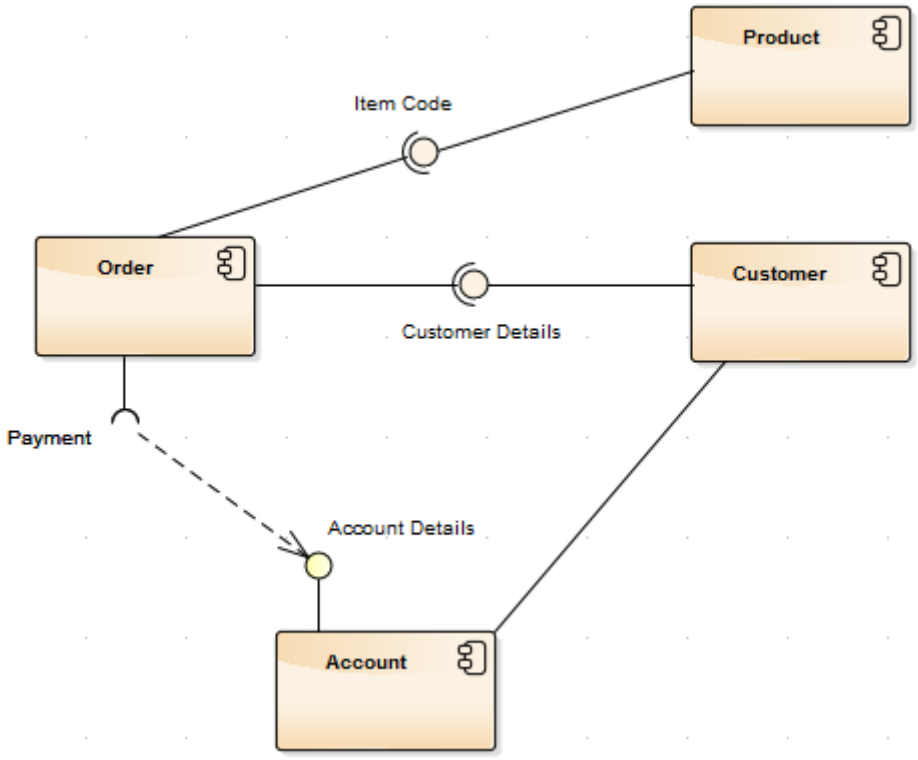
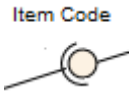


Abb.: Quelle - http://sparxsystems.com/enterprise_architect_user_guide/13.0/model_domains/componentdiagram.html

Das Bild zeigt insgesamt 4 Komponenten.

Montagestecker 

verbinden die Auftragsschnittstelle mit dem Produkt und dem Kunden. In diesem Fall benötigen die Komponenten ID-Produktinformationen und Angaben zum Kunden.

Eine einfache Linie zwischen Kundenkomponente und Kundenkonto veranschaulicht die Beziehung zwischen den Komponenten.

Objektdiagramm

Das Objektdiagramm stellt Objekte (Klasseninstanzen) und deren Beziehungen (Links - Assoziationsinstanzen) in einem Moment dar. Dieses Diagramm sieht aus wie ein Klassendiagramm und kann tatsächlich als sein spezifisches Beispiel betrachtet werden. Sie wird hauptsächlich verwendet, wenn wir Beispiele für Datenstrukturen in einem bestimmten Moment zeigen wollen.

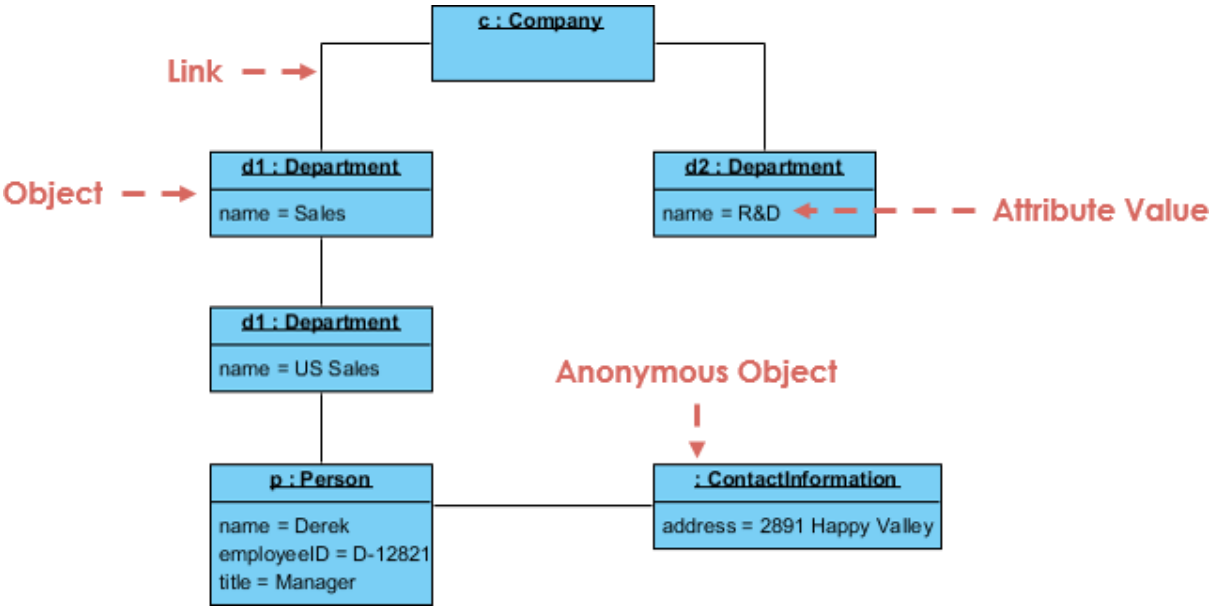


Abb.: Quelle - https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-object-diagram/

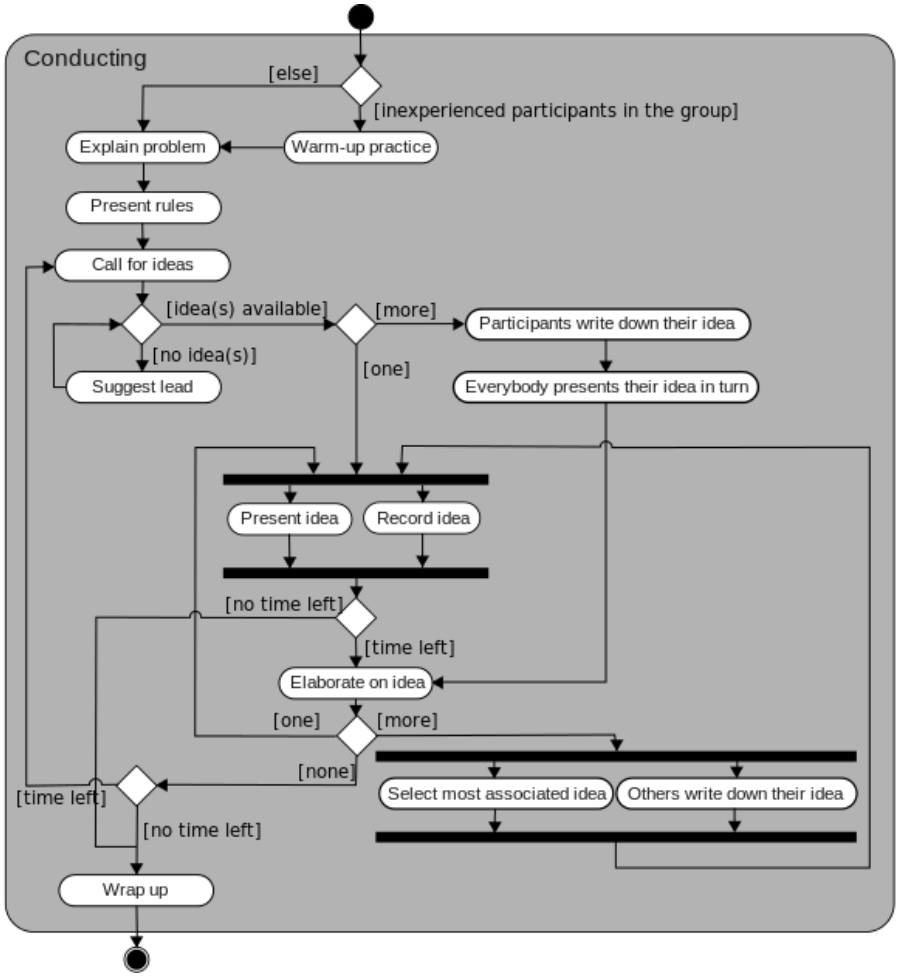
7. Verhaltensdiagramme

Diese Diagramme beschreiben einzelne Prozesse durch Aktivitäten, die ihre Zustände und Übergänge darstellen. In Anbetracht der Tatsache, dass Aktivitätsdiagramme das Verhalten des Systems beschreiben, werden Verhaltensdiagramme verwendet, um Funktionalitäten des Softwaresystems zu beschreiben.

Aktivitätsdiagramm

Dieses Diagramm ist für die Modellierung von Rechen- und Organisationsprozessen (z.B. Arbeitsprozess) konzipiert und kann zur Modellierung von Datenflüssen verwendet werden. Die Diagrammmodelle verarbeiten Prozesse mittels Aktivitäten, die ihre Zustände darstellen, und Übergängen zwischen einzelnen Zuständen. Dadurch ist es eines der Diagramme, das das Verhalten des Systems erkennen kann.

- Abgerundete Rechtecke stellen Aktionen dar.
- Diamanten stellen Entscheidungen dar.
- Balken stellen einzelne Aktivitäten dar.
- Der schwarze Kreis stellt den Anfang (Anfangsknoten) dar.
- Der umlaufende schwarze Kreis stellt das Ende (Endknoten) dar.



Sequenzdiagramme

Sequenzdiagramme zeigen Objektinteraktionen, die an einer bestimmten Systemfunktionalität beteiligt sind. Neben Objekt und Klassen zeigt es die Abfolge der Nachrichten an, die notwendig sind, um die jeweilige Funktionalität zwischen den einzelnen Objekten in Bezug auf ihre Zeitreihenfolge zu erreichen. Diese Diagramme werden häufig bei der Implementierung von Anwendungsfällen zur Logik des Entwicklungssystems verwendet. Ziel dieser Aktivität ist es, die Art und Weise vorzuschlagen, wie das Produkt in der Umsetzungsphase umgesetzt wird.

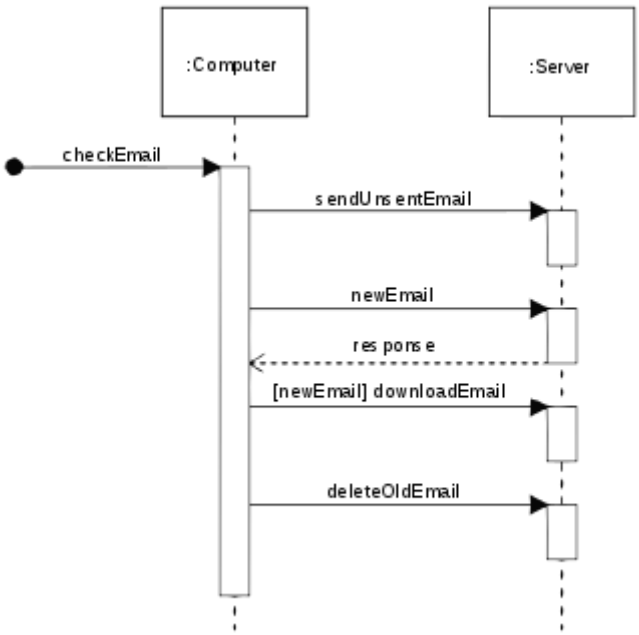


Abb.: Quelle - https://en.wikipedia.org/wiki/Sequence_diagram#/media/File:CheckEmail.svg

Die Zeitlinie im Diagramm ist vertikal (Zeitflüsse von oben nach unten). Die Platzierung der Objekte erfolgt horizontal. Unser Bild zeigt 2 Objekte (Server und Computer); die gestrichelte vertikale Linie wird auch als Lebenslinie bezeichnet. Unser Fall bezieht sich auf Objekte, die vor der Zeichnung des Diagramms existierten und auch nachträglich existieren werden. Rechtecke auf der Lebenslinie veranschaulichen eine Aktivität. Horizontale Pfeile stellen Meldungen dar. Die Pfeilrichtung zeigt den Absender und den Empfänger an. Die volle Zeile zeigt die obligatorische Nachricht, während die gestrichelte Linie eine nicht obligatorische darstellt.

Anwendungsfall-Diagramm

Es stellt in der Regel die Beziehung zwischen dem Kunden und dem System dar, wo es verschiedene Anwendungsfälle in der Kunden-System-Beziehung demonstriert, die oft mündlich durch Szenarien beschrieben wird. Um Szenarien auf Anwendungsfalldiagramme abzubilden, werden Anwendungsfall-Implementierungen angewendet. Hier soll der Algorithmus der Transkription von Szenarioeinträgen in das Diagramm behandelt werden.

So kann beispielsweise BPMN der erste Impuls für die Erstellung dieses Diagramms sein. Das Ergebnis ist eine Liste von Aktivitäten, die von der Entwicklungssoftware anstelle von z.B. Personen durchgeführt werden können. Diagrammfiguren werden als Akteure bezeichnet. Akteure sind mit relevanten Anwendungsfällen verknüpft. Die einfache Linie wird als Assoziation bezeichnet. Die horizontale Richtung zeigt weiter die Struktur der

Akteure. Der untere Akteur erbt die Eigenschaften von den Akteuren über ihm. Diese Art von Beziehung wird als Generalisierung bezeichnet. Die Include-Beziehung wird implementiert, wenn der Anwendungsfall, aus dem sich diese Beziehung entwickelt, implementiert wird.

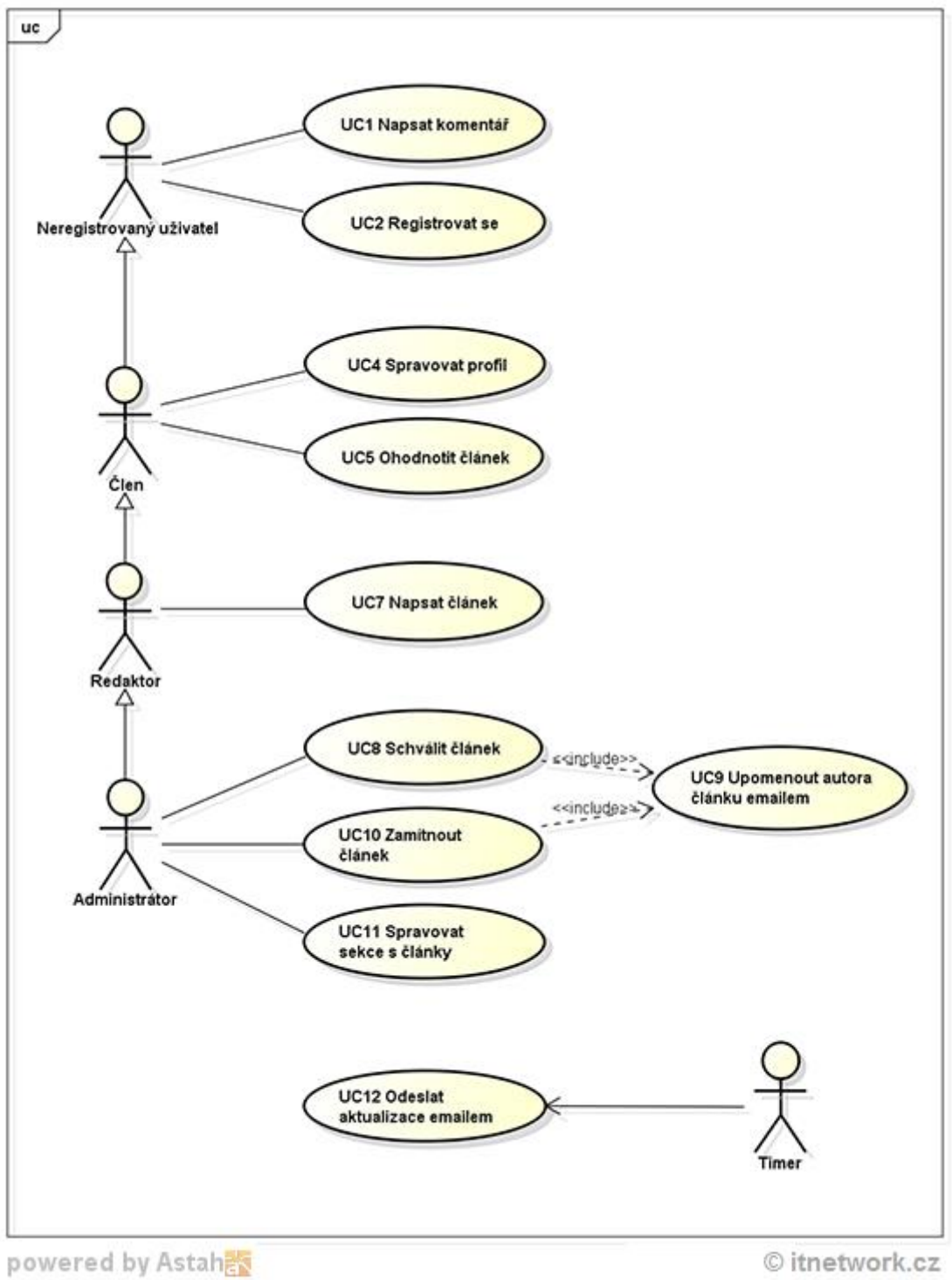


Abb.: Quelle - https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

Softwaretest und -installation
Es gibt viele Methoden des Softwaretestens, während sich jede Testmethode in der Regel auf einen anderen Testbereich konzentriert. Die einzelnen Methoden lassen sich in funktionale und nicht-funktionale Tests unterteilen. Der Funktionstest konzentriert sich auf die Erfüllung aller Anforderungen der Anwender. Die Ziele der Tests sind in der Norm ISO/IEC 12207 definiert oder können aus den Ergebnissen, die mit Hilfe von FURPS+ erzielt wurden, gewonnen werden. Jede Anforderung wird mit einer Prüfung geliefert.

Informelle Testfälle hingegen testen nicht direkt die Systemfunktionalität, sondern basieren auf normal erwarteten Operationen, die das System normalerweise durchführen kann. Diese sind wie folgt:

- Sicherheitstests - testet die Systemsicherheit
- Lasttests - Tests, bei denen ein System gefordert wird und dessen Reaktion gemessen wird.
- Usability-Tests - messen die Bedingungen, unter denen das System eingesetzt werden kann. Zum Beispiel die Verbindung der Internetgeschwindigkeit.
- Dokumentationsprüfung - Die Benutzerdokumentation wird auf ihre Verständlichkeit und Konsistenz geprüft. In der Entwicklungsdokumentation wird die Deckungshöhe getestet.

Auch Softwaresysteme werden hinsichtlich ihrer Verifikation und Validierung getestet.

- Die **Verifizierung** beantwortet die Frage, ob das entwickelte Produkt alle technischen Anforderungen erfüllt.
- Die **Validierung** hingegen beantwortet die Frage, ob die Software dem Verwendungszweck entspricht.

Wenn ein Kunde zum Beispiel ein E-Shop-System wünscht, und unser entwickeltes System viele unterhaltsame Funktionen hat, diese aber versteckt sind, so dass der Kunde sie normalerweise nicht findet, ist die Folge, dass er das Produkt nicht kauft.

Manchmal sprechen wir über Black Box und White Box Tests.

- **Black Box Testing** nähert sich dem entwickelten Softwaresystem als Black Box. Es geht also nicht um das Innere. Es untersucht die Funktionalität nur aus der Sicht des Endverbrauchers. In der Regel wird geprüft, ob die vorgegebenen Eingänge mit den erforderlichen Ausgängen übereinstimmen. Es versucht auch, ungewöhnliche oder unerwartete Eingaben zu finden, die möglicherweise nicht angemessen reagieren.

- Der **White Box Test** hingegen ist mit der Programmstruktur vertraut. Es wird geprüft, ob einzelne Programmfunktionen ordnungsgemäß funktionieren. Es ver-

sucht, jede Funktion einzeln zu testen.

Vondrák (2002) empfiehlt folgende Aktivitäten zur Bereitstellung des Softwaresystems für den Benutzer:

- Entwicklung des Endprodukts oder seiner Versionen
- Fertigstellung des Softwaresystems
- Vertrieb des Softwaresystems
- Installation des Softwaresystems beim Benutzer
- Assistenzdienste für Benutzer
- Planungsmanagement für Beta-Tests
- Migration bestehender Daten- und Softwareprodukte

8. Messung der Qualität von Softwaresystemen

Wenn es uns gelingt, die Anforderungen der Anwender/Kunden am Ende des gesamten Prozesses zu erfüllen, ist es wahrscheinlich, dass eine hochwertige Software entwickelt wurde. Dennoch ist dieser Indikator sehr subjektiv. Tatsächlich hat jeder Benutzer/Kunde unterschiedliche Erwartungen. Daher ist es notwendig, objektivere Kriterien für die Bewertung von Qualität und Nutzen von Softwaresystemen festzulegen.

Diese Kriterien werden als Metriken bezeichnet. (Es ist ein etwas unglücklicher Ausdruck da es den gleichen Namen hat wie die Verallgemeinerung des physikalischen Abstandes in der Mathematik)

Žáček (2017) definiert den Begriff der Metrik wie folgt:

Eine Metrik kann als solche definiert werden: "Eine Kennzahl ist ein klar definierter Finanzindikator oder nichtfinanzieller Indikator oder Bewertungskriterium, das zur Beurteilung des Effizienzgrades im jeweiligen Bereich des Geschäftsergebnisses und seiner effektiven Unterstützung durch IS/ICT verwendet wird." Die Gruppe von Metriken, die für einen bestimmten Zweck (d.h. bezogen auf einen bestimmten Bereich, Prozess oder ein bestimmtes Projekt) zugeordnet sind, werden als "Portfoliokennzahlen" bezeichnet.

Metriken werden in der Regel in harte und weiche Metriken unterteilt.

- Harte Kennzahlen - sind objektiv messbar, z.B. durchschnittliche Reaktionszeit, Anzahl der Fehler pro Zeiteinheit oder Garantiezeit.

- Weiche Kennzahlen - sind subjektiver und sehr schwer objektiv zu messen. Sie können auf einer Skala von typischerweise 0 - 100 in der Reihenfolge der bewerteten Produkte bewertet werden oder alternativ kann eine mündliche Bewertung durchgeführt werden.

Darüber hinaus gibt es eine Vielzahl von Systemen zur Qualitätskontrolle der Softwaresystementwicklung. Im Großen und Ganzen mögen Kunden, wenn ihre Produkte das Qualitätszertifikat haben. Tatsächlich zahlen sie gerne zusätzliches Geld dafür. Diese Systeme helfen, Fehler zu minimieren und die Effizienz des Entwicklungsprozesses zu steigern. Einige davon werden im folgenden Text kurz vorgestellt.

8.1. CMMI

Fähigkeits-Reifegradmodell (Integration). In den USA und Japan ist dieses Modell bemer-

kenswert weit verbreitet. Es ist für Entwicklungsteams konzipiert. Dieses Modell ist relativ aufwendig und kann daher als Leitfaden für die Verbesserung der Qualität des Entwicklungsteams dienen. Es besteht aus einer Reihe von Regeln und Empfehlungen, die für eine schnelle Entwicklung und Gestaltung neuer Produkte zu beachten sind. Darüber hinaus konzentriert es sich auf die Organisation, Planung und Überwachung von Entwicklungsprozessen. Sie wird durch die Aufteilung der Entwicklungsteams in fünf Reifegrade und Kapazitäten festgelegt. Auf diese Weise können sich einzelne Teams innerhalb dieser Ebenen bewegen. Die Reifegrade werden von einem gut ausgebildeten Gutachter in einem genauen Verfahren bewertet.

Die Reifegrade sind (laut Wikipedia)

- Initial: Auf dieser Ebene führen Teams diese Prozesse entweder gar nicht oder nur teilweise durch.
- Verwaltet: Das Projektmanagement ist definiert und die Aktivitäten sind geplant.
- Definiert: Die Verfahren werden definiert, dokumentiert und verwaltet.
- Quantitativ verwaltet: Produkte und Prozesse werden quantitativ verwaltet.
- Optimieren: Das Team optimiert kontinuierlich seine Aktivitäten.

Kapazitätsstufen

- Unvollständig: Einige Aktivitäten werden nicht ausgeführt.
- Durchgeführt: Es werden Aktivitäten durchgeführt.
- Verwaltet: Die Tätigkeiten werden entsprechend ihrem Management durchgeführt.
- Definiert: Die Aktivitäten werden gemäß ihrer Definition durchgeführt.

8.2. ISO 9000:2001

Im Gegensatz zu CMMI ist es nur lose definiert. Es definiert das Qualitätsmanagementsystem. Diese Norm ermöglicht es spezifischen Unternehmen, ihre Kapazitäten für die Produktion und den Vertrieb von Produkten nachzuweisen. Sie ist dabei nur lose definiert. Der Entwicklungsprozess ist daher sehr spezifisch. Aus diesem Grund wurde die Interpretation des englisch-schwedischen TickIT oder der ISO/IEC 90003 durchgeführt.

8.3. Six Sigma

Es handelt sich um eine Reihe von Techniken für das Prozessmanagement, deren Ziel es ist, seine Effizienz zu verstehen und kontinuierlich und regelmäßig zu verbessern oder die Anforderungen des Kunden durch Verständnis der Kundenbedürfnisse, Prozessanalyse, Standardisierungsmessverfahren und deren Analyse mit statistischen Methoden zu erfüllen.

Grundlegende philosophische Prinzipien dieser Methodik sind:

- Eine nachhaltige Anstrengung, um stabile und vorhersehbare Ergebnisse des Prozesses zu erzielen, ist entscheidend für den wirtschaftlichen Erfolg.
- Produktions- und Geschäftsprozesse haben Eigenschaften, die definiert, bewertet, analysiert, verbessert und kontrolliert werden können.
- Um eine ständige Qualitätsverbesserung zu erreichen, muss das gesamte Unternehmen, einschließlich des Top-Managements, eingebunden werden.
- Zu den wichtigsten Techniken gehören beispielsweise der DMAIC-Zyklus - er dient dem Verständnis und der Steuerung von Prozessen.

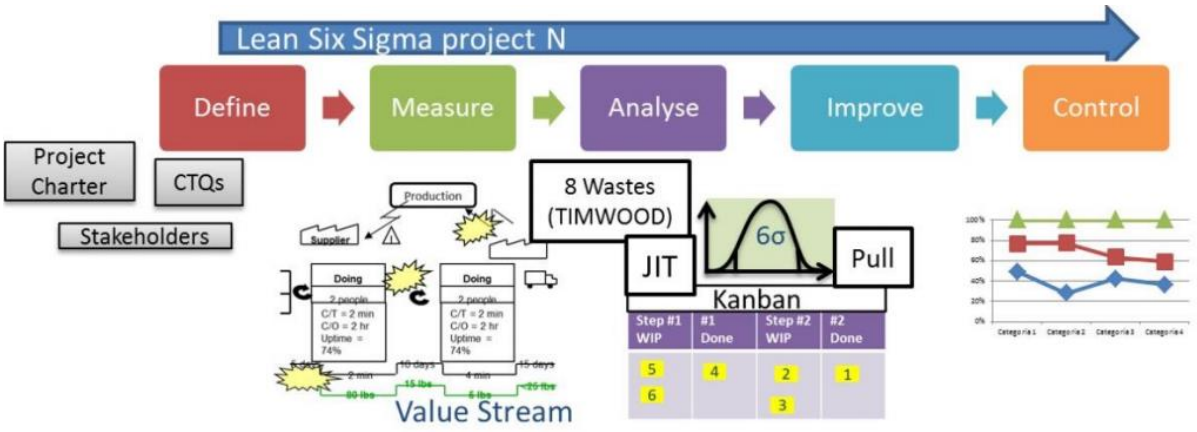


Abb.: *Quelle - http://www1.osu.cz/~zacek/infs2/02.pdf*

DMAIC besteht aus 5 Phasen:

- **Definieren** - Definieren eines tatsächlichen und idealen Prozesses. Es schlägt einen Weg vom eigentlichen zum idealen Prozess vor.

- **Messen** - Messung des aktuellen Prozesses. Es bedeutet zum Beispiel die durchschnittliche Wartezeit, die uns mit "Hard Data" über die "objektive Realität" versorgt.

- **Analyse** - die Messergebnisse werden statistisch ausgewertet, um ein Arbeitsmodell für ein bestimmtes Projekt zu erstellen; alternativ versucht die Analyse herauszufinden, welche Teile des Prozesses verbessert werden müssen.

- **Verbessern** - die Umsetzung von Gedanken aus der vorherigen Phase. Zu den wichtigsten Zielen gehören die Zufriedenheit des Kunden, die Steigerung der Effi-

zienz und die Senkung der Kosten.

- Kontrolle - wenn eine Änderung erfolgreich eingeführt wurde, muss sie standardisiert und kontrolliert werden, ob sie eine Verbesserung gebracht hat oder nicht.

Diese Phase wurde ursprünglich von der Motorola Company eingebracht. Diese Phase wird heute unter anderem von Honeywell, HP, Texas Instruments, NASA und anderen Unternehmen genutzt. Sie basiert auf angewandten statistischen Verfahren, die mit qualitativen Werkzeugen ausgestattet sind.

9. Wartung und Betrieb von Softwaresystemen

Bisher haben wir uns nur mit der Softwareentwicklung beschäftigt. Dennoch markiert die Softwareentwicklung erst den Beginn des Software-Lebenszyklus. Es ist vor allem der Betrieb, der den entscheidenden Teil ausmacht. Genauer gesagt, wenn wir ein leistungsfähiges Softwaresystem entwickeln würden, das nach einem Monat seines Betriebs aufgrund seiner schlechten Wartung regelmäßig ausfallen würde, würde der Kunde sicherlich das Gefühl haben, dass das gesamte System schlecht entwickelt ist. Daher ist es notwendig, eine regelmäßige Wartung durchzuführen, damit die Software effektiv funktioniert.

ITIL konzentriert sich auf die Techniken der Wartung und des Betriebs von Informationstechnologien. Es handelt sich um eine Reihe von Konzepten und Techniken für eine effektivere Nutzung von IT-Services. IT-Services umfassen IT-Systeme, deren Hauptziel es ist, Geschäftsprozesse zu stimulieren (d.h. die Unternehmen müssen ihre Arbeit richtig machen).

ITIL ist eigentlich ein Band von Buchpublikationen, die wertvolle Erfahrungen im Bereich des Service Managements von Informationstechnologien enthalten. Damit wird ein Rahmen für das IT-Service-Management geschaffen, der uns sagt, wann und was zu tun ist. Das ITIL-Konzept ist eigentlich sehr einfach; oder genauer gesagt, warum wir den gesamten Prozess von Anfang an gestalten und entwickeln sollten, wenn viele andere Unternehmen ihn bereits eingeführt und regelmäßig verbessert haben.

ITIL verwendet einen proaktiven Ansatz. Es geht darum Probleme im Voraus und nicht rückwirkend festzustellen. Wenn ein Problem bereits aufgetreten ist, versucht es, es durch aktive Erkennung herauszufinden, für welche es korrekte Verfahren festlegt. Der gesamte Prozess wird regelmäßig gesteuert, überwacht, bewertet und ständig verbessert. Alle diese Prozesse sollen dem Kunden einen Mehrwert bieten. Die Terminologie kann jedoch in einigen Fällen, in denen Unternehmen unterschiedliche Fachbegriffe verwenden, ein Problem darstellen. ITIL strebt dabei eine Standardisierung der Terminologie an. Diese Verfahren sind plattformneutral etabliert.

Der größte Vorteil dieser Ansätze ist wahrscheinlich ihre Effizienz, dank derer beispielsweise die Dauer von IT-System-Ausfällen reduziert werden kann.

Aus diesem Grund wurden zwei grundlegende Prozesse Service Support und Service Delivery eingeführt. Innerhalb dieser Prozesse wurde eine Kundenkontaktstelle - Service Desk - eingeführt. Dieser Punkt sammelt die Anforderungen der Kunden oder Benutzer. Außerdem werden IT-Prozesse erfasst, so dass sie reagieren und die Anforderungen erfüllen können. Es bietet auch grundlegende IT-Unterstützung.

Im Problemfall findet der Incident Management Prozess statt, der versucht, unangenehme Folgen für Kunden bei Problemen mit IT-Systemen zu minimieren. Eines der wichtigsten Kriterien ist die schnelle Lösung des Problems.

Darüber hinaus wird das Vorfallmanagement durch das Problem Management unterstützt, das das Register zur Problembehandlung verwaltet. Gleichzeitig analysiert es bestehende Probleme und deren Charakter und führt gegebenenfalls strukturelle Veränderungen der IT-Systeme ein.

Einzelne Änderungen werden im Change Management erfasst. Release-Management plant und verwaltet individuelle Änderungen von IT-Services (Release)

ITIL Version 3 besteht aus 5 Teilen und einem Einführungsbuch; (Žáček, 2017) beschreibt sie wie folgt:

- Service-Strategie - befasst sich mit der Harmonisierung von Business und IT, Management-Strategie von IT-Services, Planung.
- Service Design - befasst sich mit IT-Services, Prozessplanung (Erstellung und Pflege von IT-Architekturen und -Verfahren).
- Service Transition - überträgt IT-Services in die Geschäftsumgebung.
- Servicebetrieb - liefert und verwaltet Prozessaktivitäten, Anwendungsmanagement, Änderungen, Betrieb und Kennzahlen.
- Kontinuierliche Serviceverbesserung - befasst sich mit IT-Services, Verbesserungskompetenzen, Methoden, Praktiken und Metriken.

10. Literatur

Eysenck, M. W., & Keane, M. T. (2008). Kognitivní psychologie. Praha: Academia.

Nolen-Hoeksema, S. (2012). Psychologie Atkinsonové a Hilgarda (Vyd. 3., přeprac.). Praha: Portál.

Sklenář, V. (2007). SOFTWAROVÉ INŽENÝRSTVÍ [Online]. Retrieved June 29, 2018, from https://phoenix.inf.upol.cz/esf/ucebni/syspro.pdf

Softwarové inženýrství [Online]. (2001-). In Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation. Retrieved from https://cs.wikipedia.org/wiki/Softwarov%C3%A9_in%C5%BEen%C3%BDrstv%C3%AD#Softwarov%C3%A1_krize

Plháková, A. (2004). Učebnice obecné psychologie. Praha: Academia.

Rational Unified Process [Online]. (2001-). In Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation. Retrieved from https://en.wikipedia.org/wiki/Rational_Unified_Process

Rychta, A. (2011). Softwarové inženýrství [Online]. Retrieved June 29, 2018, from http://www.ksi.mff.cuni.cz/~richta/NSWI026/NSWI026-1-Uvod.pdf

Říčan, J. (2016). Používané metakognitivní strategie žáků pátých tříd ve specifické doméně čtení (Disertační práce). Praha.

US Department of Justice (2003). INFORMATION RESOURCES MANAGEMENT Chapter 1. Introduction.

Vondrák, I. (2002). Úvod do softwarového inženýrství [Online]. Retrieved June 29, 2018, from http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf

WHITE, Stephen A. Business Process Modeling Notation [online]. [cit. 2018-06-26]. Dostupné z: https://is.muni.cz/el/1433/jaro2014/PV165/um/46771256/pr_06_bpmn.pdf

Žáček, J. (2017). SOFTWAROVÉ INŽENÝRSTVÍ [Online]. Retrieved June 29, 2018, from http://www1.osu.cz/~zacek/sweng/skripta_sweng.pdf

GURNDLAGEN DER WEBANWENDUNGEN

1. Kommunikation, Netzwerke, Protokolle

1.1. Kommunikationsprotokolle

Im Kontext von Computernetzwerken sehen wir oft Kommunikationsprotokolle. Sie definieren genau die Art und Weise, wie die Kommunikation eine bestimmte Funktion über alle Ebenen hinweg erfüllen kann. Es gibt Protokolle für das Senden von Daten, das Erstellen sicherer Kanäle und das Suchen nach IP-Adressen, die dem Domänennamen entsprechen, oder das Zustellen von E-Mails, etc. Das Protokoll ist beiden Kommunikationspartnern bekannt und beschreibt genau, welcher Inhalt, welche Reihenfolge und welches Timing für die Übertragung verwendet wird. Jede Abweichung von dieser strukturierten Kommunikation kann als Fehler interpretiert werden.

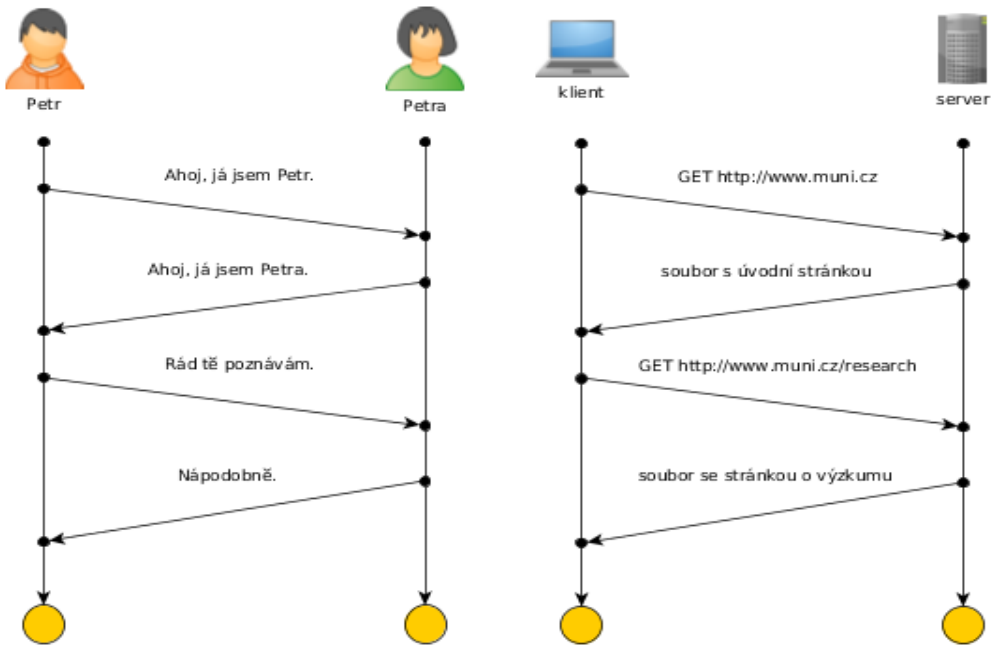


Abb.: Darstellung des Kommunikationsprotokolls

1.2. TCP/IP

Das wesentliche Prinzip der Computernetzwerkarchitektur ist die Aufteilung der Kommunikation in Schichten durch Abstraktion. Jede Schicht ist dafür verantwortlich, die Übertragung zu beschreiben, die die Anwendungsebene über physikalische Datenverbindungskommunikation startet. Das TCP/IP-Netzwerkmodell ist der Eckpfeiler aller bestehenden Netzwerke sowie des gesamten Internets. Der Name leitet sich von zwei wichtigen Protokollen ab, die das Datenrouting und den Transport zwischen den Knoten sicherstellen. Das IP-Protokoll beschreibt die Adressierung von Knoten, paketweise zerlegte Daten und deren Weiterleitung innerhalb des Netzwerks.

Die Kommunikation zwischen den gleichen Schichten zweier verschiedener Systeme wird durch das Kommunikationsprotokoll unter Verwendung einer Datenverbindung gesteuert, die von der benachbarten unteren Schicht hergestellt wird. Die Architektur erlaubt es, Protokolle einer Schicht auszutauschen, ohne die anderen zu beeinflussen. Die TCP/IP-Architektur ist in vier Schichten unterteilt (im Gegensatz zum OSI-Referenzmodell mit sieben Schichten):

- Anwendungsschicht
- Transportschicht
- Internet-Schicht
- Netzwerkschnittstelle

Abb.: TCP/IP-Netzwerkmodellschichten

TCP ist ein Transportprotokoll, das einen "verknüpften" Dienst auf IP-Übertragung aufbaut: Es stellt die Steuerung der erfolgreichen Übertragung und potenziellen Weiterleitung von fehlenden oder beschädigten Teilen sicher und wählt eine optimale Übertragungsgeschwindigkeit.

Die TCP/IP-Protokollfamilie, die derzeit für die absolute Mehrheit der Netzwerkkommu-

nikation verwendet wird, wurde zwischen den 70er und 80er Jahren mit dem Ziel entwickelt, ein robusteres Modell der Netzwerkkommunikation zu schaffen, das bis zu einem gewissen Grad in der Lage ist, Verluste von Teilen des Netzwerks zu bewältigen. Die entscheidende (und damals revolutionäre) Idee ist die Paketvermittlung: Daten werden nicht als kontinuierlicher Strom übertragen, sondern in separaten Blöcken (Paketen) und jeder Netzwerkknoten allein bestimmt, in welche Richtung Pakete gesendet und weitergeleitet werden.

In der Praxis basiert diese Idee auf einer Reihe von Protokollen, die die notwendigen Funktionen implementieren. Protokolle werden typischerweise in mehrere Ebenen (Layer) unterteilt. Anstelle des abstrakten ISO/OSI-Modells, das mit sieben Schichten arbeitet, wird für die TCP/IP-Interpretation meist ein vereinfachtes Fünf-Schichten-Modell verwendet (einige Protokolle im TCP/IP-Modell implementieren die Funktion mehrerer Schichten des ISO/OSI-Modells).

1.2.1. ANWENDUNGSSCHICHT

Die Anwendungsschicht ist die oberste Schicht, die sich auf die Anwendungsprotokolldaten der einzelnen Netzwerkdienste bezieht. Es beinhaltet Netzwerkanwendungsprotokolle: elektronische Post, HTTP (Websites), DNS (Domain-Service) und andere. Es gibt eine Vielzahl solcher Protokolle, z.B. HTTP, SMTP, FTP, NTP sind die am häufigsten bekannten. Für TCP/IP sind das Daten, die an einen Zielempfänger übertragen werden müssen. Die Implementierung ist bis zu den unteren Schichten möglich.

1.2.2. PHYSIKALISCHE SCHICHT

Die physikalische Schicht ist die unterste Schicht im Modell. Im Gegensatz zu den oberen Schichten ist es keine Softwareschicht (Protokoll) und bezieht sich auf ein bestimmtes physikalisches Medium zur Datenübertragung. Ein Beispiel kann ein Twisted Pair, die Verkabelung in den meisten lokalen Ethernet-Netzwerken, Koaxialkabel, Glasfaser oder eine Telefonleitung sein. Das Medium muss nicht materiell sein - z.B. bei drahtlosen Netzwerken im Mikrowellenband (Wi-Fi, Breezenet) oder optischen Freiraumverbindungen (FSO).

1.2.3. DATENVERBINDUNGSSCHICHT

Die Datenverbindungsschicht ist die niedrigste der Softwareschichten. Es ist das niedrigste Kommunikationsprotokoll, das für die Datenübertragung auf einem physikalischen Medium verwendet wird. Dieses Protokoll ist meist eng mit einem bestimmten Medium verbunden, aber diese Korrespondenz muss nicht 1:1 sein, z.B. ist Ethernet nicht nur auf Twisted-Pair-Verkabelung implementiert, sondern Sie sehen auch Implementierungen mit Koaxialverkabelung oder Glasfasern. Ein weiteres Beispiel für ein Da-

tenverbindungsschichtprotokoll ist PPP, das für die Implementierung von DFÜ-Verbindungen oder seriellen Computerverbindungen verwendet wird. Das wichtige Merkmal von Data Link Layer Protokollen ist, dass sie sich nur mit der Kommunikation zwischen direkt verbundenen Knoten (daher der Name) befassen.

1.2.4. INTERNET-SCHICHT

Die Internet-Schicht ist für die globale Adressierung und Weiterleitung zuständig, in der Praxis am häufigsten durch ein IP-Protokoll (in zwei Versionen IPv4 und IPv6) implementiert. Obwohl es Adressen in Link-Layer-Protokollen und bei Ethernet-MAC-Adressen gibt, sind sie global eindeutig (oder sollten es zumindest sein), können sie jedoch nicht für das Paketrouting verwendet werden, da sie das Ziel nicht anzeigen. IP-Protokolladressen (IP-Adressen) sind jedoch hierarchisch aufgebaut, so dass die Delegation einzelner Bereiche die Netzwerktopologie widerspiegelt. Daher können Sie anhand der Ziel-IP-Adresse erkennen, wohin Sie ein Paket weiterleiten sollen, d.h. mindestens den folgenden Hop auf der Route. Zusätzlich zu dieser Grundfunktion ermöglicht das IP-Protokoll beispielsweise die Fragmentierung (Zerlegung zu großer Pakete in mehrere kleinere Datagramme) oder die Paketbezeichnung nach Art des Dienstes (ToS).

1.2.5. TRANSPORTSCHICHT

UDP und TCP sind die am häufigsten verwendeten Protokolle der Transportschicht. UDP (User Datagram Protocol) ist ein minimal nachrichtenorientiertes Transportschichtprotokoll, das nur das Konzept des Ports als spezifische Prozessadresse (besser gesagt als Socket) innerhalb des Zielknotens einführt. Aber UDP ist immer noch ein zustandsloses Protokoll (keine Verbindung im wahrsten Sinne des Wortes), das sich nicht mit dem Problem der verlorenen Pakete oder ihrer Reihenfolge beschäftigt. Dennoch wird es häufig verwendet, weil es einfach und kostengünstig ist, insbesondere dort, wo diese Fragen nicht behandelt werden müssen.

Der umgekehrte Ansatz wird durch TCP (Transmission Control Protocol) repräsentiert. Im Gegensatz zu UDP stellt es Verbindungen zwischen zwei Ports auf den Endknoten her. Aus Sicht von Client-Anwendungen ist das Verhalten dieser Verbindung vergleichbar mit einer Kommunikationsleitung zwischen zwei Prozessen (aber im Gegensatz zur Leitung ist TCP eine Zwei-Wege-Verbindung). Es wird sichergestellt, dass ein von der einen Seite versandter Stream (Bytefolge) auf der anderen Seite im gleichen Format empfangen wird. Das TCP-Protokoll ermöglicht das Erkennen und erneute Senden von verlorenen Daten sowie die Neuanordnung von Daten aus Paketen, die in der falschen Reihenfolge ankommen. Da TCP-Transportprotokolle einen relativ hohen Komfort für die Anwendungsschicht gewährleisten, werden sie derzeit für den Großteil der Kommunikation verwendet. Der Nachteil ist ein höherer Overhead und eine relativ geringe Reaktion auf Ausfälle, weshalb UDP für bestimmte Zwecke (z.B. die meisten DNS-Abfragen oder VoIP)

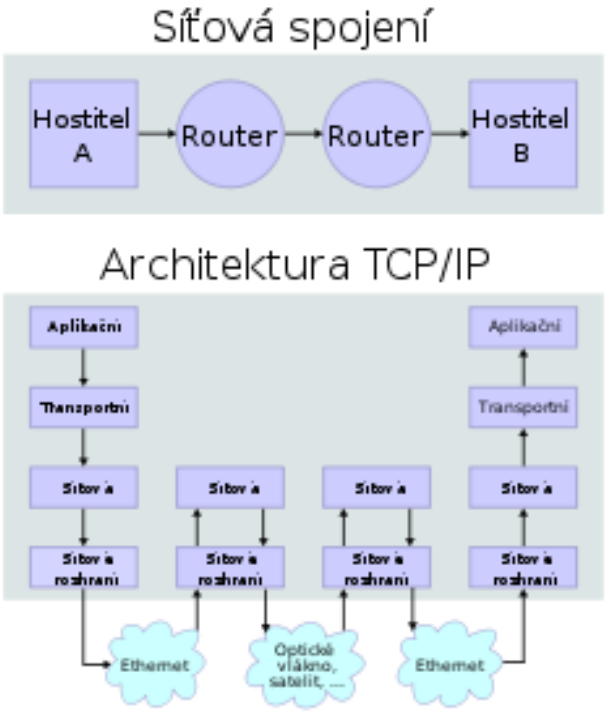
bevorzugt wird.

1.3. ICMP

ICMP (Internet Control Message Protocol) geht etwas über die Layerstruktur hinaus. Pakete (Nachrichten) dieses Protokolls werden direkt über ein IP-Protokoll übertragen. ICMP wird jedoch nicht als Transportprotokoll betrachtet, da es nicht für die Übertragung von Anwendungsdaten verwendet wird. Dieses Protokoll wird für Diagnose- und Servicezwecke verwendet. Die Beispiele für ICMP-Anwendungen sind paketbestimmte, nicht erreichbare Nachrichten, ICMP-Echo und Echo-Antwortpakete oder einige Diensttypen von Nachrichten (Redirect).

1.3.1. PAKETGRÖßE

Die maximale (theoretische) Größe des IP-Pakets beträgt 65535 B, aber die Datenverbindungsschicht ist oft ein limitierender Faktor. Da die meisten Pakete mindestens einmal über das Ethernet (oder das Äquivalent) laufen, wird die Größe der Pakete oft nach ihrer Grenze (1536 B) ausgewählt, daher ist der typischste Wert 1500 B. Aber natürlich ist dies der Maximalwert und die Pakete sind sehr oft viel kleiner, insbesondere für interaktive Anwendungen. Die Größe der IP- und TCP-Header beträgt 20-60 B (typischerweise nahe der unteren Grenze), UDP- und ICMP-Header sind 8 B und Ethernet-Header sind 14 B (zusätzlich ist 2 B am Ende des Pakets die Prüfsumme).



Die TCP/IP-Schichten, die den Datentransfer zwischen zwei Hosts über zwei Router ermöglichen.

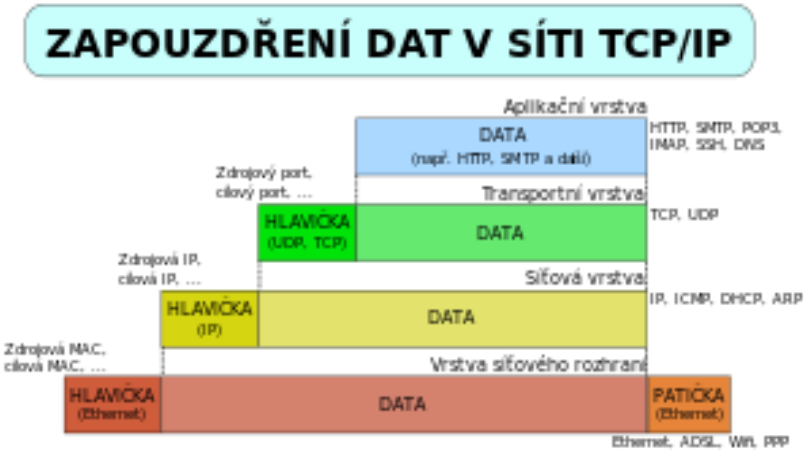


Abb.: Übersicht über die Kapselung von Anwendungsdaten auf TCP/IP-Schichten.

Angesichts der Problemkomplexität wird die Netzwerkkommunikation in sogenannte Schichten unterteilt, die die Hierarchie der Aktivitäten darstellen. Der Informationsaustausch zwischen den Schichten ist genau definiert. Jede Schicht verwendet Dienste der unteren Schichten und stellt Dienste für die oberen Schichten bereit.

Das Internetprotokoll ist das Basisprotokoll der Internetschicht und des gesamten Internets. Es generiert Datagramme basierend auf Netzwerk-IP-Adressen, die in ihren Headern enthalten sind. Es stellt den verbindungslosen Netzwerkdienst für die oberen Schichten zur Verfügung.

Am weitesten verbreitet ist derzeit das IP-Protokoll Version 4. Die neue Version 6, die den Mangel an Adressen in IPv4 und Sicherheitsprobleme behebt und andere Eigenschaften des IP-Protokolls verbessert, wird nur von ein paar Prozent der weltweit mit dem Internet verbundenen Geräte genutzt, aber die Zahl steigt rasant an.

1.4. IPv4

- Internet-Protokoll Version 4
- 32-Bit-Adressen
- App. 4 Milliarden IP-Adressen, jetzt nicht mehr ausreichend
- Format: xxx.xxx.xxx.xxx.xxx.xxx wobei xxx eine beliebige Zahl ist 0 - 255 (8 Bit)

1.5. IPv6

- Internetprotokoll Version 6
- 128 Bit Adressen
- Sicherheitsunterstützung
- Unterstützung für mobile Geräte
- QoS - Quality of Service Funktion
- Paketfragmentierung - Verteilung
- Nicht kompatibel mit IPv4

Das Address Resolution Protocol wird verwendet, um nach einer physikalischen MAC-Adresse mit einer bekannten IP-Adresse zu suchen. Bei Bedarf sendet das Protokoll ein Datagramm mit den gesuchten IP-Adressinformationen und adressiert diese an alle Stationen im Netzwerk. Der Knoten mit der gesuchten Adresse antwortet und gibt seine MAC-Adresse ein. Wenn sich der gesuchte Knoten nicht im gleichen Segment befindet, antwortet der jeweilige Router und gibt seine Adresse zurück.

1.6. ICMP

Das Internet Control Message Protocol wird für die Übertragung von Steuernachrichten in Bezug auf Fehlerzustände und besondere Übertragungsbedingungen verwendet. Es wird z.B. für Computerverfügbarkeitstests im Ping-Programm oder für die Verfolgung von Paketrouten zu einem anderen Knoten durch das Traceroute-Programm verwendet.

1.7. TCP

Das Transmission Control Protocol schafft eine virtuelle Verbindung zwischen den Endanwendungen, d.h. eine zuverlässige Datenübertragung. Die Protokollmerkmale sind:

- Zuverlässiger Transportservice, der alle Daten verlustfrei und in der richtigen Reihenfolge an den Empfänger liefert.
- Verbindungsdienst mit Phasen des Verbindungsaufbaus, der Datenübertragung und des Verbindungsabbaus.
- Transparente Übertragung beliebiger Daten.
- Vollduplexverbindung, parallele Zweiwege-Datenübertragung.
- Unterscheidung zwischen Anwendungen, die Ports verwenden.

1.8. UDP

Das User Datagram Protocol bietet einen unzuverlässigen Transportdienst für Anwendungen, die keine Zuverlässigkeit des TCP-Protokolls benötigen. Es fehlt die Verbindungsaufbau- und -abbauphase und bereits das erste UDP-Segment enthält Anwendungsdaten.

1.9. SCTP

Ein zuverlässiges Protokoll für die Übertragung von Datagrammen in mehreren Streams. Sie wird in erster Linie für die Telekommunikation eingesetzt. Es hat einige zusätzliche Funktionen, die TCP nicht bietet:

- Multihoming - der kommunizierende Knoten kann mehrere IP-Adressen haben.
- Aufspaltung des Datenstroms in Datagramme.
- Verwendung mehrerer Datenströme - zur Reduzierung der Blockierung der Kommunikation aufgrund eines fehlenden Datenblocks, der bei TCP auftreten kann.
- Routenauswahl und -verfolgung - Verwendung einer Alternative, wenn die Hauptadresse Verfügbarkeitsprobleme hat.

2. Sprachen für die Präsentation von Webinhalten

2.1. Einführung in HTML

HTML ist eine einfache Auszeichnungssprache für die Erstellung von Websites, die nur gewöhnliche Textdateien sind, die einen Text und mehrere HTML-Tags enthalten, die die Bedeutung und das Aussehen der einzelnen Website-Teile bestimmen.

Neben html gibt es noch mehrere andere Sprachen, die für die Erstellung von WWW-Seiten verwendet werden: css, php, javascript, etc.

HTML ist eine Grundlage, wenn Sie etwas über CSS-Stile (CSS-Stile bestimmen das Aussehen von Webseiten) und später über PHP (bezogen auf die Programmierung) erfahren möchten.

2.1.1. GESCHICHTE VON HTML

Web erschien 1989 und wurde seither kontinuierlich entwickelt. Derzeit ist HTML 5 als Entwickler tätig.

2.1.2. ERSTELLUNG VON HTML-SEITEN

Wir können einen einfachen Texteditor, Notepad oder spezielle Editoren mit HTML-Codeunterstützung verwenden. Diese Programme heben die Syntax hervor und ermöglichen es uns, zwischen einem Code und einer Vorschau zu wechseln. Die Arbeit mit einem solchen Editor ist viel effizienter als die Arbeit mit einem herkömmlichen Notizblock.

2.1.3. IN HTML VERWENDETE BEGRIFFE.

Tag ist eine grundlegende HTML-Markierung, geschrieben als <tag>.

Attribut wird innerhalb eines Tags geschrieben und setzt seine Eigenschaft. Es ist geschrieben als: <tag attribute="value">.

Element Schreiben einer Überschrift: <h1>heading</h1>.

Erste Seite

HTML-Seiten sind nur gängige Textdateien mit Tags.

Tags

Tags werden zwischen Klammern < > geschrieben, einige von ihnen sind gepaart, andere nicht.

Schreiben eines nicht gepaarten Tags:

```
<tag>
```

Schreiben eines gepaarten Tags:

```
<tag> einiger Text </tag>
```

Im Falle eines gepaarten Tags ist es wichtig, einen Schrägstrich zu schreiben, da der Explorer sonst nicht verstehen würde.

Attribute

Attribute werden direkt in den Tag geschrieben.

```
<tag attribute=”value”>
<tag attribute=”value”>Gemeinsames Tag mit attribute.</tag>
```

Es ist verboten Tags zu überkreuzen

Tags können ineinander verschachtelt werden, dürfen sich aber nicht überkreuzen.

```
<b><i>fett kursiv</b></i>
```

Richtige Schreibweise:

```

```

```
<b><i>fett kursiv</i></b>
```

Größe der Zeichen

In html spielt die Größe der Zeichen keine Rolle, daher ist es möglich, sowohl <TAG> als auch <tag> zu schreiben, während in xhtml, der neuesten Version von html, weder Tags noch Attribute in Großbuchstaben geschrieben werden können.

In der URL ist es notwendig, die Größe der Zeichen zu beachten, d.h. FILE.html ist nicht dasselbe wie file.html.

2.1.4. HTML

HTML steht für Hypertext Markup Language. Hypertext-Markup-Sprache war Entwickler von SGML und wurde zur am weitesten verbreiteten Sprache für die Erstellung von Webseiten. In der Vergangenheit waren die am häufigsten verwendeten Versionen HTML 2.0, HTML 3.2, HTML 4.01 und HTML 5. Von HTML wurde auch XHTML (extended hypertext markup language) als Anwendung von XML entwickelt, was meiner Meinung nach nicht von höherer Bedeutung ist. Im Jahr 2010 wurde HTML 5 in Betracht gezogen; die meisten seiner Innovationen wurden bereits eingesetzt.

Struktur einer html-Datei

Die am häufigsten verwendete "Vorlage" einer Seite:

```
<!DOCTYPE HTML>
<html>
  <head>
      <meta charset=”windows-1250”>
      <title>Name</title>Name</title>
  </head>
  <body>

Seitentext

  </body>
</html>
```

2.2. CSS

2.2.1. GESCHICHTE VON CSS

CSS erschien um 1997. Es ist eine Sammlung von Methoden zur grafischen Gestaltung von Webseiten. Die Abkürzung steht für Cascading Style Sheets. Sie werden als Kaskadierung bezeichnet, da Definitionen eines Styles überlagert werden können, aber nur die letzte ist gültig. Dies ist jedoch im Moment nicht wichtig.

Es wurden auch CSS 2, verbesserte und ausgefeiltere Stilformen entwickelt, die jedoch im am häufigsten verwendeten Internet Explorer nicht gut funktionieren.

Wann Sie CSS verwenden sollten

Im Jahr 2015 könnte man sagen, dass das gesamte Web mit CSS formatiert ist. Aus dem bisher verwendeten HTML wurden nur noch Fettheit und Kursivschrift verwendet. Daher ist es gut, einige Kenntnisse in CSS zu haben, wenn Sie Webseiten erstellen wollen. Zunächst einmal ist es notwendig zu wissen, wie HTML funktioniert. Ohne Grundkenntnisse ist es nicht empfehlenswert, mit CSS zu arbeiten. CSS ist zu untersuchen, wenn:

- Sie möchten formatierte Websites haben - Farben, Rechtfertigung des Textes, Spaltenlayout, etc.
- Sie schreiben oft Texte für das Internet, ohne Zeit mit der Formatierung zu verlieren,
- Sie beschäftigen sich mit Skripten, insbesondere mit Javascripten.
- Web mit vielen Seiten verwalten (oder planen), die ein ähnliches Design haben sollen

2.2.2. ANWENDUNGEN VON CSS

Es gibt drei Möglichkeiten, CSS zu verwenden

Der Stil kann auf drei Arten bezeichnet sein (siehe die Beispiele unten). Es genügt, mindestens eine der folgenden Methoden zu kennen:

Im Quelltext für ein formatiertes Element über das Attribut style="....". Dies wird als direkter Stil bezeichnet. Es ist völlig ungeeignet.

Mit Hilfe von "Style Sheet" im Kopf der Seite. Style Sheet ist eine Liste von Styles, die Informationen über die Formatierung liefert (z.B. grüne Überschriften). In einer Seite wird ein Style Sheet zwischen den Tags <style> und </style> geschrieben.
Mit Hilfe des externen Stylesheets - es ist eine Datei *.css, mit der die Seite mit einem

Tag <link> verknüpft ist. Die Datei enthält ein Stylesheet. Der Hauptvorteil ist, dass viele Seiten mit einer Datei verknüpft werden können und alle Seiten ein ähnliches Design haben.

Beispiele

Erstellen Sie ein Paragraf in roter Schrift mit CSS. Es stehen drei Möglichkeiten zur Verfügung:

Direkter Stil

In der Quelle wird diese Absatzerklärung geschrieben:

```
<p style="color: red">Dieser Absatz wird rot sein.</p>
```

Erklärung: <p> ist eine Markierung, die einen Absatz bezeichnet. Das Attribut "style" ist ein allgemeines Attribut, das für jedes Element verwendet werden kann.

Style Sheet

In der Dokumentenüberschrift wird ein Stylesheet zwischen den Tags <style></style> geschrieben:

```
<style>
        p {Farbe: rot}
</style>
```

Absätze werden im Textkörper der Seite geschrieben:

```
<p>Dieser Absatz wird rot angezeigt. </p>
<p>Dieser Absatz wird auch rot sein, da alle Absätze rot sein
werden.</p>
```

Wenn wir wollen, dass nur einige Absätze rot sind, verwenden Sie "classes" und "identifier".

Durch externe CSS-Datei

Es wird eine Datei erstellt, die z.B. *styly.css* heißt. Es wird nur den folgenden Text enthalten:

```
p {color: red}
```

In der Überschrift eines html-Dokuments, das wir nach Stil ändern wollen, muss der folgende Link geschrieben werden:

```
<link rel="stylesheet" type="text/css" href="styly.css">
```

Im Textkörper des Dokuments werden alle Absätze rot dargestellt.

2.2.3. SYNTAXE

Wie Sie vielleicht bemerkt haben, ist CSS kein Teil von HTML, daher sind sie anders geschrieben. Wenn die folgende Tabelle zu theoretisch erscheint, beachten Sie bitte nur die Beispiele im unteren Teil.

Direct style

```
<tag style="characteristics">styled element</tag>
```

Style sheet

```
<style>
        tag {characteristics
        2nd tag {characteristics}
</style>
```

Vereinfachte Charakteristiken:

```
characteristics: value; 2nd characteristics: 2nd value
```

Allgemeine Charakteristiken:

```
Characteristics: value [, value2] [; another characteristics]
```

Beispiele

Direct style

```
<p style="color: red;">text of red paragraph</p>
```

Style sheet

```
<style>
        p {color: red}
        body {background-color: yellow;}
</style>
```

Einfache Schreibweise der Charakteristiken:

```
color: red
```

Komplexe Schreibweise der Charakteristiken:

```
font-family: Arial, Arial CE, sans-serif; color: red;
```

Es ist zu beachten, wenn Anführungszeichen, Doppelpunkte, geschweifte Klammern, Semikolons und Kommas verwendet werden. Beispiel für eine korrekte Schreibweise:

```
h2 {color: green; background-color: yellow}
```

Leerzeichen und Zeilenenden spielen keine Rolle, sie können hinzugefügt und weggelassen werden. Die Zeichengröße spielt dabei keine Rolle. Es steht eine Liste der Merkmale und deren Werte zur Verfügung.
Der Explorer ignoriert die Werte, die er nicht erkennt.

In Style Sheets werden Kommentare ähnlich wie in Java zwischen */* a */* Zwei Schrägstriche funktionieren nicht.

Beispiel mit einer Überschrift

Es ist ganz einfach in Stylesheets oder externen CSS-Dateien.

```
<style>
        h1 {color: green;}
        h2 {color: blue;}
</style>
```

Auf diese Weise enthält das gesamte Dokument grüne Überschriften der ersten Ebene und blaue Überschriften der zweiten Ebene, wobei jedoch nur angenommen wird, dass die Tags *<h1>* und *<h2>* zum Schreiben eines Textes verwendet wurden. Mit anderen Worten, Stylesheets können nur in gut strukturierten Texten verwendet werden.

2.2.4. CSS-STILE

CSS-Stile sind kaskadierend, sie werden verwendet, um einen Stil einer Webseite zu erstellen (Farbe, Schriftart, Schriftgröße). Mit CSS kann eine Datei das Design des gesamten Webs beeinflussen.

Veraltete Methoden

Vor CSS-Stilen wurde für einen Webseiten-Stil das Element *<font face="schriftart" color="farbe" size="größe">* verwendet, das nicht mehr verwendet wird. Gegenüber CSS hat es folgende Nachteile:

- Wenn Sie den Stil eines Textes oft geändert haben, erschien dieser Tag sehr oft im Quellcode, was die Seite verlangsamte.
- Es ermöglicht, nur Schriftart, Farbe und Größe zu ändern. *<font face="schriftart" color="farbe" size="größe">*

2.3. DHTML

Vielleicht sind Sie schon auf die Abkürzung DHTML oder Dynamic HTML gestoßen. Diese Sprache wird fast ausschließlich aus JavaScript, VBScript (Sprache mit ähnlichen Eigenschaften wie JavaScript) und CSS-Stilen erstellt. Diese Sprache nutzt die Vorteile von HTML, JavaScript und CSS und schafft so ein perfektes Design und gut aussehende Seiten.

2.3.1. CSS-STILE UND -KLASSEN, BEZEICHNER UND STIL

CSS - Cascading Style Sheets - Cascading Styles wurden von Microsoft erstmals 1996 in Internet Explorer 3.0 implementiert. Der CSS-Stil entfernt <Font> vollständig und führt <style> ein. Mit Hilfe von CSS-Stilen ist es möglich, Farbe, Schriftart, Größe und viele andere zu definieren (Box, Unterstreichung, Kühnheit, Welligkeit, Display, Aufzählungen, Ränder...).

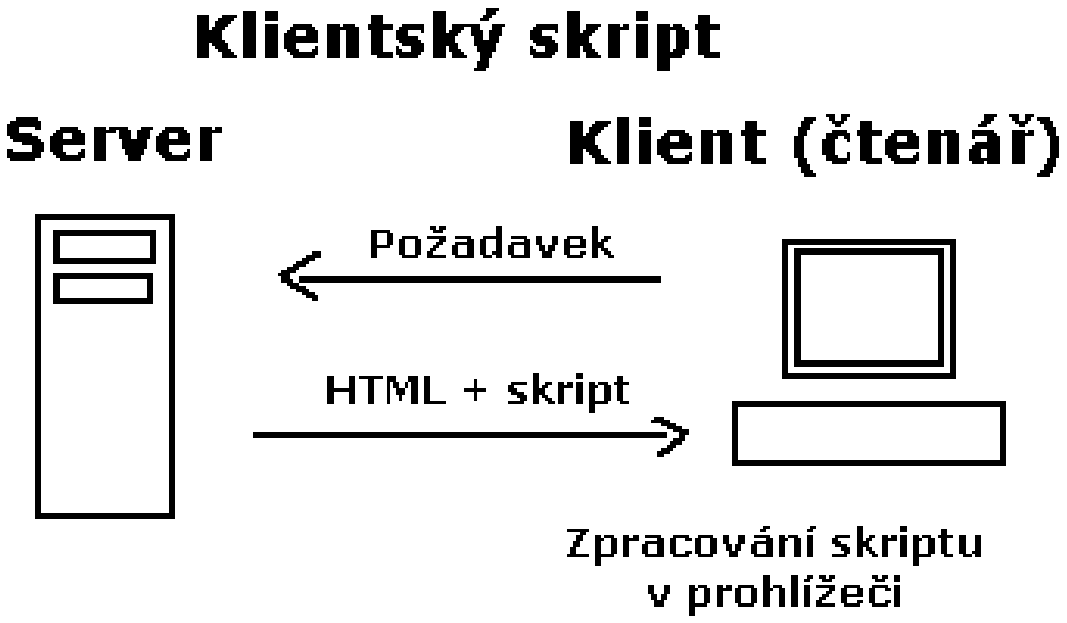
CSS-Stile werden hauptsächlich über Klassen und Identifikatoren angewendet. Diese ermöglichen es, einen CSS-Stil nur mit einem Attribut zu erstellen und der Benutzer muss einen Code nicht zehnmal wiederholen. Darüber hinaus ist es auch möglich, den Stil der Elemente (h l, Tabelle, etc.) über Selektoren zu definieren. So wird beispielsweise jedes Element <input> immer einen roten Text haben - dies ist mit einer CSS-Zeile möglich.

3. Client-seitige Logik - JavaScript

3.1. Was ist JavaScript

JavaScript ist eine Programmiersprache, die auf Websites verwendet wird. Es wird direkt in HTML-Code geschrieben, was aufgrund seiner Einfachheit ein großer Vorteil ist.

JavaScript ist ein Client-Skript. Das bedeutet, dass das Programm mit einer Seite an den Client (in den Explorer) gesendet und dort ausgeführt wird (im Gegensatz zu Serverskripten, die auf einem Server ausgeführt werden und der Client nur die Ergebnisse erhält).



Legende: klientský skript - Kundenskript, klient (čtenář) - Kunde (Leser), požadavek - Anforderung, zpracování skriptu v prohlížeči - Verarbeitungsskript im Explorer

Es gibt auch andere Sprachen von Client-Skripten, z.B. VBScript. Es wird jedoch so selten verwendet, dass, wenn man "Skripte" erwähnt, dies meist "JavaScripte" bedeutet.

JavaScript ist nicht Java

JavaScript wird oft mit Java verwechselt. Java ist eine separate Programmiersprache. Es hat nur eine ähnliche Syntax wie JavaScript.

Erforderliche Fähigkeiten

- HTML, Grundlagen von HTML mindestens
- Grundlagen der Programmierung

3.1.1. DIE SPRACHMERKMALE

JavaScript ist eine Sprache

- interpretiert - es muss nicht kompiliert werden.
- beanstandet - es verwendet das Objekt eines Explorers und eingebaute Objekte.
- abhängig vom Explorer - funktioniert aber in den meisten Explorern
- Groß-/Kleinschreibung beachten - die Größe der Zeichen im EINTRAG ist entscheidend.
- seine Syntax ist ähnlich wie bei C, Java und dergleichen.
- Einschränkungen der Sprache
- JavaScript funktioniert nur innerhalb eines Explorers.
- Benutzer können JavaScript blockieren
- Es gibt verschiedene Versionen der Sprache und der Explorer, was zu häufigen Fehlern führt.
- es kann nicht auf Dateien (außer Cookies) oder andere Systemobjekte zugreifen.
- Es können keine Daten (außer Cookies) gespeichert werden.
- Dies macht es nur zu einer Sekundärsprache, die nur für HTML-Seiten gilt.

Wie man mit JavaScript umgeht

Nach der Beherrschung der Grundlagen empfiehlt es sich, Skripte auf anderen Websites zu beachten. Die meisten Skripte werden direkt in den Quellcode der Webseiten geschrieben, so dass es möglich ist, sie zu kopieren (einige Codes befinden sich in externen Dateien, aber auch diese können heruntergeladen werden.

3.1.2. ERLÄUTERUNG DES SKRIPTS

Das Skript wird in HTML zwischen den Tags `<script>` und `</script>` geschrieben. Alles, was zwischen den Tags geschrieben wird, ist ein Programm, das in der Sprache Javascript geschrieben wurde.

Das Beispiel verwendet den Befehl `document.write()`. Dies ermöglicht das normale Schreiben in den Dokumentenstrom. Der geschriebene Text wird sofort im Explorer angezeigt.

Normaler Text muss zwischen Anführungszeichen geschrieben werden (im Gegensatz zu Variablen). Eine Zeile darf nicht zwischen den Anführungszeichen umgebrochen werden. Jeder JavaScript-Befehl wird mit einem Semikolon oder einem Zeilenumbruch abgeschlossen.

Wie man das erste Skript erstellt

Alles, was in JavaScript erstellt wird, wird als Skript bezeichnet. Es kann auf der Seite platziert werden oder es ist möglich, einen Link darauf zu erstellen. In diesem Fall wird die Seite in der JavaScript-Seite hochgeladen. Separate Dateien, die in JavaScript geschrieben wurden, haben die Endungen .js oder .jse. Die Erweiterung .js ist gebräuchlicher. Das Einzige, was Sie für die Erstellung eines Skripts benötigen, ist ein Quelltexteditor (PSPad, Texteditor oder ein beliebiger HTML-Editor). Zum Surfen benötigen Sie einen Explorer (mindestens Internet Explorer und Mozilla Firefox, damit Sie Skripte in diesen am häufigsten verwendeten Explorern überprüfen können).

Einfügen eines Skripts in eine Seite

Das Skript wird zwischen den Tags `<script>` und `</script>` geschrieben. Sie können zwischen dem Abschnitt "Körper" oder "Kopf" eingefügt werden (abhängig vom Zweck eines Skripts).

```
<html>
  <head>
        ...
        <script type=”text/javascript”>
              javascript script body ..
        </script>
        ...
  </head>
  <body>

document body
  <script type=”text/javascript”>
        .javascript script body
  </script>

document body
  </body>
</html>
```

Tag <script>

Die Syntax des Tags <script> lautet wie folgt:

```
<script type=”text/javascript” src=”url of external file”>
  <!--
       javascript script content
  //-->
<script>
```

Der Attributtyp bezeichnet einen Typ eines Skripts (im Falle von JavaScript "text/javascript"). Da es Explorer gibt, die JavaScript nicht verstehen, wird empfohlen, <!-- am Anfang des Skripts und //--> am Ende des Skripts zu schreiben, sonst würde der Explorer das Skript als normalen Text schreiben (jetzt betrachtet er den Text als Kommentar und zeigt ihn nicht an).

3.1.3. SKRIPTERSTELLUNG

In JavaScript ist es wichtig, zwischen Groß- und Kleinschreibung zu unterscheiden, daher ist document.write nicht dasselbe wie DOCUMENT.write. Diese Regel muss beachtet werden, sonst funktioniert das Skript nicht.

JavaScript ist eine plattformübergreifende, objektorientierte Skriptsprache, deren Autor Brendan Eich von der Firma Netscape ist.

Nun wird JavaScript in der Regel als interpretierte Programmiersprache für WWW-Seiten verwendet, die oft direkt in einen HTML-Seitencode eingefügt wird. Es steuert in der Regel verschiedene interaktive Elemente GUI oder erstellt Animationen und Bildeffekte.

JavaScript war ursprünglich ein Handelsname einer Implementierung von Netscape, die ursprünglich unter dem Namen Mocha, später LiveScript, entwickelt wurde. Es wurde im Dezember 1995 mit der Firma Sun Microsystems als Ergänzung zu den Sprachen HTML und Java angekündigt. Für die Version der Firma Microsoft wird der Name JScript verwendet. Dies wird von der Plattform.NET unterstützt.

Im Gegensatz zu anderen interpretierten Programmiersprachen (z.B. PHP und ASP), die noch vor dem Download aus dem Internet serverseitig gestartet werden, wird ein Programm in JavaScript in der Regel nach dem Herunterladen einer WWW-Seite aus dem Internet (auf Client-Seite) gestartet. Dies impliziert bestimmte Sicherheitseinschränkungen. Beispielsweise kann JavaScript nicht mit Dateien arbeiten, da es sonst die Privatsphäre des Benutzers gefährden könnte.

JavaScript kann auch auf der Seite eines Servers eingesetzt werden. Die erste serverseitige Implementierung von JavaScript war LiveWire der 1996 gegründeten Firma Netscape. Heutzutage gibt es mehr Möglichkeiten, einschließlich Open-Source-Implementierung Rhinola auf Basis von Rhino, gcj, Node.js und Apache.

Neben DHTML wird JavaScript zum Schreiben von Erweiterungen für viele Anwendungen verwendet, z.B. Adobe Acrobat.

JavaScript kann auch in den Betriebssystemen Windows mit dem Programm Windows Script Host verwendet werden und ersetzt so Batch-Dateien MS-DOS.

4. Web-Architektur

4.1. Webarchitektur-Design

Wir beginnen nie damit, ein Web mit grafischem Design zu erstellen. Zunächst werden die Ziele der einzelnen Seiten definiert und deren Platzierung im Web festgelegt.

Warum ist es wichtig?

Dank des Designs der Webarchitektur ist es möglich, sich vorzustellen, wie das resultierende Web aussehen wird und wie seine Funktionen aussehen werden.

Auf diese Weise ist es möglich, potenzielle Fehler zu identifizieren und zu beheben und so Geld zu sparen, das für die spätere Reparatur programmierter Anwendungen erforderlich wäre.

Webstruktur-Design

Im Strukturdesign müssen wir alle Webseiten, deren gegenseitige Verknüpfungen und übliche Benutzerszenarien definieren. Es ist notwendig, sicherzustellen, dass die Informationen angemessen strukturiert und einfach zu bedienen sind und zu Zielerreichungen führen.

Web-Navigationsdesign

Eine geeignete Navigation ist ein wichtiger Bestandteil eines Webs. Es ermöglicht einem Besucher, schnell zu finden, was er sucht. Diesem Aspekt ist bei der Gestaltung eines Webs besondere Aufmerksamkeit zu schenken.

Drahtgittermodellierung

Sobald wir den Inhalt des Webs kennen, kann mit der Wireframe-Modellierung begonnen werden, die die Grundlage für das grafische Design des Webs bildet.

4.2. Programmierung von Webanwendungen

Webanwendung bedeutet in der Regel ein serverseitiges Skript (ein Code, der die Programmfunktion sicherstellt). Es ist oft mit einer Datenbank verbunden, einem System, das Daten von Webanwendungen speichert (vereinfacht gesagt, kann man sich eine Datenbank als MS Excel-Datei vorstellen). Die Skriptausgabe ist eine Webseite, die zur Anzeige an den Explorer übergeben wird.

Legende: schéma webové aplikace - Webapplikationsgraph, Skript - Skript, www stránka - www page, databáze – Datenbank

Die Aufgabe von Webanwendungen besteht vor allem darin, die Interaktion der Internetpräsentation mit ihren Besuchern zu erhöhen oder die Webadministration zu erleichtern, d.h. repetitive Arbeit bei der Erstellung von WWW-Seiten zu ersparen.

Abhängig von den Anforderungen an die Funktionalität kann eine Webanwendung aus nur wenigen Zeilen Code bestehen (z.B. beim Versenden von Kontaktformularen), aber es gibt auch Webanwendungen mit Tausenden von Zeilen.

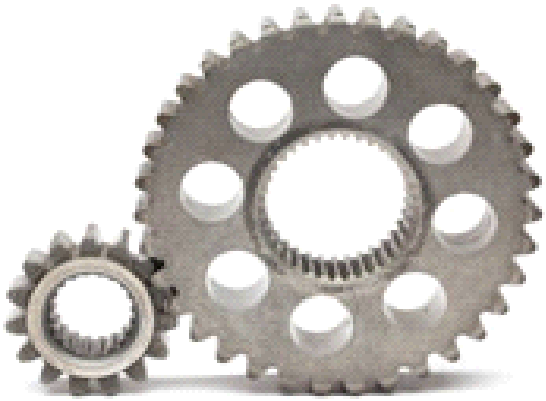
Komplexere Webanwendungen sind oft mit anderer Software innerhalb eines Unternehmens verknüpft, z.B. mit Bestellsystemen, Buchhaltungsprogrammen, etc. Dies ermöglicht es, Kosten für die menschliche Arbeit zu sparen. Es ist auch möglich, die Anwendung an Online-Zahlungssysteme anzubinden.

Beispiele für einfache Webanwendungen

- Kontaktformular
- Gästebuch
- Diskussionen oder Chat
- Kataloge und Preislisten
- Wörterbücher
- Bannersysteme

Bei größeren Präsentationen ist es besser, die gesamte Präsentation dynamisch anzugehen, z.B. über Vorlagen oder den Einsatz des Content Management Systems. Eine spezifische Form einer solchen Anwendung ist der Internet-Shop. Aufgrund ihrer vielen Vorteile werden Weblogs sowie verschiedene Intranets und Extranets immer beliebter. Alle gehören zu den speziellen Arten von Webanwendungen.

Bei der Programmierung aller Webanwendungen sind die folgenden Aspekte zu berücksichtigen:



Sicherheit - ist eine Priorität für jede Webanwendung, da immer die Gefahr besteht, dass Daten verloren gehen oder zerstört werden. Es besteht auch die Gefahr, dass die Daten oder Informationen von einer unbefugten Person gestohlen werden oder dass der Webserver mittels einer Anwendung gehackt wird (Gefahr des Verlustes des Firmenimages).

Nutzung der verfügbaren Ressourcen - bei der Programmierung jeder Webanwendung versuchen wir, fertige Codestücke aus anderen Ressourcen zu verwenden; zu diesem Zweck besitzen wir ein umfangreiches Skriptarchiv. Auf diese Weise kann etwas Arbeit, die für die Entwicklung der Anwendung notwendig ist, und damit auch Geld und die gesamte Zeit für die Realisierung des Auftrags eingespart werden.

Skalierbarkeit - wenn sich herausstellt, dass die Webanwendung in der Praxis gut funktioniert, beginnt sie in der Regel modifiziert, verbessert und erweitert zu werden. Werden diese Änderungen bei der Anwendungsgestaltung berücksichtigt, ist die Einbindung viel einfacher und kostengünstiger. Aus diesem Grund werden die komplexesten Webanwendungen modular angegangen.

Geschwindigkeit - langsame Webanwendungen sind nicht sehr anwendbar, sie sind auch ein Problem für Explorer. Daher versuchen wir, alle Skripte so zu optimieren, dass sie schnell sind, und deshalb wird auch empfohlen, eine fertige Anwendung auf unseren Servern zu installieren. Da auch die neuesten Entwicklungstools installiert sind, die auf vielen Servern für kommerzielles Webhosting fehlen, ist die Entwicklung der Webanwendung auch schneller und kostengünstiger.

Die maximale Belastung der Webanwendung ist ein Begriff, der mit hohem Traffic verbunden ist. Wenn ein Web von einer höheren Anzahl von Besuchern besucht wird (z.B. wenn Sie erwarten, dass ein Produkt präsentiert wird), ist der Server oft nicht in der Lage, sie zu bedienen (wir sagen, dass der Server "abgestürzt" ist). Die Fähigkeit einer Webanwendung, der hohen Belastung standzuhalten, erfordert die Auswahl geeigneter Tools, Datenbankoptimierung, Berechnungen und andere spezielle Techniken wie das Pre-Caching. Aufgrund der Erfahrung können wir sagen, dass der geringe Lastwiderstand ein Beweis für die geringe Leistungsfähigkeit der Programme ist, die die Weban-

wendung erstellt haben.
Testen - Vor dem Start einer Webanwendung sollen alle ihre Funktionen auf einem Entwicklungsserver getestet werden. Dieser hat die gleiche Konfiguration wie der eigentliche Server, was eine Reduzierung möglicher Probleme während der eigentlichen Inbetriebnahme ermöglicht.

4.3. Technologie unserer Webanwendungen

Die derzeit am weitesten verbreitete Skriptsprache für die Programmierung von Webanwendungen ist PHP und die Datenbank MySQL. Diese Kombination mit einem Webserver (Programm) Apache wird Triade genannt. Es hat sich als nützlich für seine Flexibilität erwiesen. Weitere Vorteile dieses Systems sind die Zugänglichkeit von Funktionen und Codefragmenten sowie die kontinuierliche Weiterentwicklung dieser Programme.

Wenn es Gründe für eine Webanwendung oder eine Vereinfachung ihrer Entwicklung gibt, werden andere Programmiersprachen verwendet, z.B. Perl, Python oder die Datenbank PostgreSQL. Jeder von ihnen ist für die spezifischen Anforderungen einer bestimmten Webanwendung geeignet.

Der größte Vorteil aller genannten Technologien ist ihre Einbindung in Open Source. Es bedeutet, dass sie frei sind (was einer der wesentlichen Faktoren für ihre Popularität und Verbreitung ist).

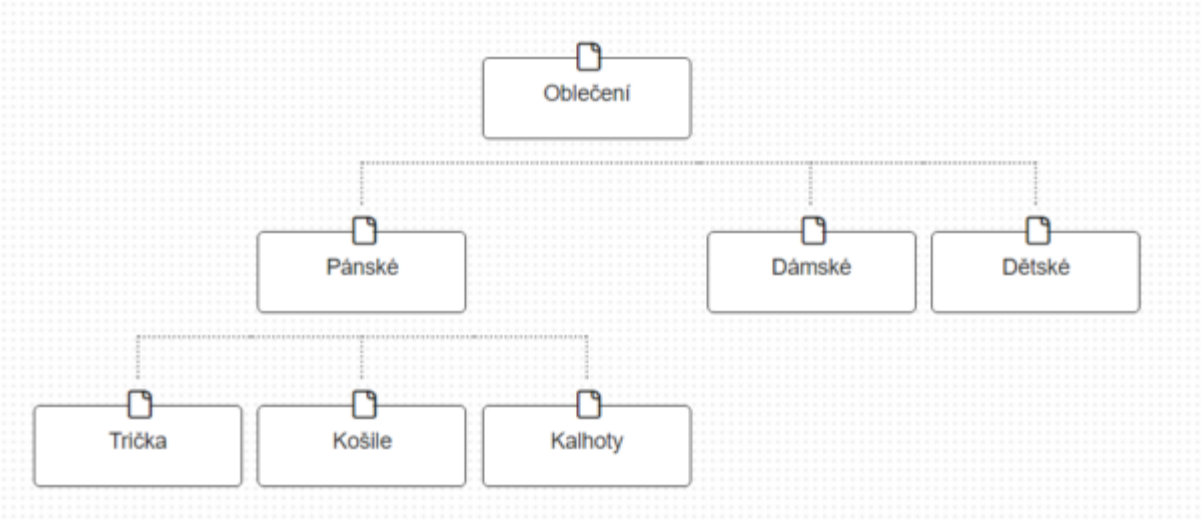
4.4. Was ist eine Webarchitektur?

Die Informations(inhalt)webarchitektur ist eine Methode, um die Webinformationen in ein logisches Ganzes zu bringen.

Es ist klar, dass die Informationen nicht auf der gleichen Seite sein werden, aber das Web wird mehr Seiten enthalten, die in verschiedenen Ebenen angeordnet sind, von der allgemeinen bis zu den Details.

Beispiel: Eine Web-Selling-Kleidung ist wie folgt angeordnet: Männer Kleidung > Hemden > Kurzarm. Die Art und Weise, wie die Informationen innerhalb eines Webs angeordnet sind, wird als Webarchitektur bezeichnet.

Weitere Informationen finden Sie in der folgenden Abbildung:



Legende: oblečení - Kleidung, Pánské - Männer, dámské - Frauen, dětské - Kinder, trička - T-Shirts, Košile - Hemden, Kalhoty – Hosen

Warum man sich mit Webarchitektur beschäftigen sollte

Eine gute Webarchitektur ist aus den folgenden Gründen wichtig:

- Es ermöglicht einem Benutzer, sein Ziel (den gewünschten Webinhalt) zu erreichen.
- Es schafft eine Logik des gesamten Webs, die sowohl von Nutzern als auch von Forschern und Suchmaschinen wahrgenommen und genutzt wird.
- Es ermöglicht die Anordnung der Webinhalte so, dass sie in Suchmaschinen leicht zu finden sind.
- Bei der Vorbereitung eines neuen Webs oder E-Shops wird die inhaltliche (Informations-)Architektur eines Webs innerhalb des grundlegenden Webdesigns behandelt (vor dem Grafikdesign und der Realisierung des Webs - siehe 4 Schritte eines professionellen Webdesigns).
- Eine gute Webarchitektur ermöglicht es, alle wichtigen Themen richtig anzuordnen. Dies spiegelt sich in einer einfacheren Orientierung sowie in einer besseren Nachvollziehbarkeit durch gezielte Keywords in Suchmaschinen wider. Wie bereits bekannt ist, ist der Traffic von Suchmaschinen kostenlos und hoch relevant.

4.4.1. WIE MAN EINE PERFEKTE ARCHITEKTUR SCHAFFT

Innerhalb von MD Webdesign wird die Webarchitektur auf Basis der Keyword-Analyse erstellt. Das bedeutet, herauszufinden, nach welchen Keywords die Nutzer in den Suchmaschinen von Google und Seznam suchen, sie zu analysieren und dann zu kategorisieren, um einen besseren Einblick zu erhalten.

Wenn wir durch ein bestimmtes Keyword nachvollziehbar sein wollen, ist es empfehlenswert, eine Webseite zum Thema Keyword und Keyword selbst zu haben.

Die Keyword-Analyse ermöglicht es, herauszufinden, woran die Nutzer der Suchmaschinen interessiert sind. Die Einbindung dieser Themen (Keywords) in die Webarchitektur erhöht die Chancen auf bessere Positionen bei der Suche erheblich.

Beim Aufbau der Informationsarchitektur wird auch der Inhalt der einzelnen Seiten beschrieben. Die Gestaltung der Webinhalte erfolgt auf der Grundlage der Analyse von Kunden / Klienten in Form einer sogenannten Person (weitere Informationen zu Personen siehe Wikipedia). Dadurch ist es möglich, ein Web zu entwerfen, das den Erwartungen der konkreten Kunden entspricht. Und wir sind in der Lage, die benötigten Informationen bereitzustellen.

Erstellung von Webseiten: beginnend mit der Informationsarchitektur

Informationsarchitektur ist ein Plan zum Sortieren von Informationen im Web. Für ein gutes Funktionieren der Webseiten muss der Schaffung einer klaren Informationsarchitektur große Aufmerksamkeit geschenkt werden.

Als Architekt, der an einem Hausdesign arbeitet, arbeitet er mit Raum, Licht und Formen, während der Informationsarchitekt mit Informationen, Strukturen und Prioritäten arbeitet.

Generell muss eine gut gestaltete Informationsarchitektur auch in der Textform verständlich sein. Die Verwendung von Farben oder Pfeilen hilft nicht, wenn die Struktur der Informationsseite nicht verständlich ist.

Die Hauptgründe für die Verwirrung des Webs sind: die verwendete Terminologie, eine Vereinbarung, die Fachkenntnisse erfordert, eine unlogische Anordnung von Informationen.

Um die oben genannten Probleme zu vermeiden, muss das Design der Informationsarchitektur die folgenden Schritte umfassen:

- Identifizierung der Zielgruppe
- Sammeln von Informationen
- Gruppierung von Informationen
- Priorisierung
- Erstellung einer Informationsarchitektur

Die einzelnen Schritte:

1. Identifizierung einer Zielgruppe

   Die Zielgruppe wird vor dem Entwurf der Informationsarchitektur identifiziert, da sich die Sichtweise auf die veröffentlichten Informationen dramatisch ändert.

   Im Falle von Websites der öffentlichen Verwaltung ist der Zielnutzer fast jeder; daher wird es schwierig sein, das Wissen oder die Gewohnheiten zu definieren, die die meisten Benutzer teilen werden. Wir können jedoch davon ausgehen, welches Wissen die Benutzer NICHT haben werden.

   Die Praxis zeigt, dass die Nutzer, die von Städten und Gemeinden aus auf die Websites zugreifen, in der Regel den Unterschied zwischen den Begriffen "Stadt" und "Behörde" nicht kennen. Sie kennen die Terminologie nicht und es ist für sie schwierig, die Organisationsstruktur einer Behörde zu verstehen. Die Situation ist noch schlimmer, da die gleiche Agenda in der Regel von verschiedenen Abteilungen in verschiedenen Städten verwaltet wird.

   Daraus folgt, dass die interne Struktur einer Behörde kein gutes Beispiel für die Schaffung einer der Öffentlichkeit präsentierte Informationsarchitektur ist.

2. Sammeln von Informationen

   Der zweite Schritt bei der Gestaltung der Informationsstruktur besteht darin, festzulegen, nach welchen Informationen der Besucher im Web suchen wird. Behörden sind verpflichtet, bestimmte Informationen zu veröffentlichen, aber es sind nicht alle Informationen, die die Website enthalten soll.

   Die Webseiten der öffentlichen Verwaltung enthalten in der Regel eine Vielzahl von Informationen, die von Sitzungsprotokollen, Verordnungen, Budgets, Beschlüssen, Vorschriften bis hin zu aktuellen Ereignissen in der Gemeinde und einer Liste von Kultur- oder Sportveranstaltungen reichen.

   Ein Teil dieses Schrittes ist auch eine Entscheidung darüber, welche Informatio-

nen NICHT im Web veröffentlicht werden. Es ist immer zu überlegen, welche Informationen der Web-Betreiber regelmäßig aktualisieren kann. Die Probleme mit veralteten Informationen können beim Design der Informationsarchitektur vermieden werden.

3. Gruppierung von Informationen

   Wenn wir wissen, welche Informationen das Web präsentieren soll, werden thematische Einheiten geschaffen. Diese Einheiten müssen alle im vorherigen Schritt gesammelten Informationen enthalten, möglichst ohne Überschneidungen.

   Zu diesem Zeitpunkt muss mindestens ein Vertreter der Öffentlichkeit in den Prozess einbezogen werden. Als Büroangestellter sind Sie sich der Struktur des Büros und des Managementprozesses bewusst. Für die Öffentlichkeit ist das anders; es ist nicht praktikabel, von einem Besucher zu erwarten, dass er sich der inneren Struktur des Büros in einem Web bewusst ist.

   Der Besucher geht "von oben" durch das Web: Er geht von einer allgemeinen Ebene aus und spezifiziert seine Anfrage schrittweise mit weiteren Klicks, bis er die gewünschten Informationen findet. Beim Entwurf der Informationsarchitektur erfolgt der Prozess jedoch "von unten" - Informationen werden sortiert und in Gruppen und Einheiten eingeteilt. Diese Diskrepanz kann für einen Besucher irreführend sein, da er die Webinhalte nicht kennt und nicht weiß, ob die Informationen im Web vorhanden sind. Daher ist es unerlässlich, die Verständlichkeit der Informationsstruktur "von oben" zu testen.

4. Priorisierung

   Bisher genügten der gesunde Menschenverstand und das Testen mit den Anwendern. Die Priorisierung im Design der Informationsarchitektur ist jedoch mit der Analyse verbunden. Daher ist es notwendig, über konkrete Zahlen zu verfügen, z.B. die Analyse der meistgesuchten Begriffe im Web. Die Prioritäten können sich auch im Laufe der Zeit ändern, z.B. im Sommer, es gibt häufige Anfragen zu Schwimmbädern etc., während im Winter die Nutzer z.B. an den Schneeverhältnissen interessiert sind.

   Es gibt viele Methoden der Priorisierung: z.B. das Platzieren der Informationen oben in der Struktur, das Platzieren auf der Titelseite oder der Hauptseite der thematischen Einheit, das Hervorheben durch Kontrast, Farben, Größe, etc.

   Beispielsweise sollen Seiten, die sich auf Siedlungsabfälle konzentrieren, Informationen über den aktuellen Preis und die Fälligkeit der Gebühr enthalten, ohne den Besucher zu zwingen, die pdf-Dateien mit einem Hinweis zu lesen. Da es sich um häufig gesuchte Informationen handelt, können sie auch mit einer größeren

Schriftart hervorgehoben werden.

5. Erstellen einer Informationsarchitektur

   Beim Aufbau einer Informationsarchitektur kann ein geeignetes Werkzeug verwendet werden, aber ein Werkzeug selbst reicht nicht aus. Die Informationsarchitektur ähnelt einem Baum.

5. PHP – basics

PHP ist eine Programmiersprache, die auf der Seite des Servers arbeitet. PHP ermöglicht das Speichern und Ändern von Website-Daten. Die ursprüngliche Bedeutung war Personal Home Page. Es wurde 1996 gegründet und hat viele Veränderungen erfahren. Heutzutage steht die Abkürzung für Hypertest Preprocessor.

PHP-Sprache

PHP ist eine der am häufigsten verwendeten Programmiersprachen für die Erstellung von Webanwendungen. PHP ist eine Server- und Serverseite zum Erzeugen eines HTML/XHTML-Seitencodes, der später an den Explorer gesendet wird (im Gegensatz zu clientseitigem JavaScript, das erst nach der Anzeige im Explorer funktioniert).

Der Hauptvorteil von PHP ist seine Unabhängigkeit von der Plattform (Windows, Linux, Unix....). Weitere Vorteile sind ein breites Anwendungsspektrum. PHP kann mit Dateien und vielen verschiedenen Datenbanken arbeiten, es kann Grafiken generieren und bearbeiten, Mails senden und empfangen, PDF erstellen, alle wichtigen Internetprotokolle unterstützen.

Da PHP eine relativ freie Syntax hat (wie es geschrieben ist), ist es leicht zu erlernen, besonders wenn Sie Erfahrung mit anderen Programmiersprachen haben. Zusammen mit dem Webserver Apache und der Datenbank MySQL bildet sie die so genannte Triade, drei Programme, die am häufigsten für die Erstellung von Webseiten verwendet werden. Dies bringt einen weiteren Vorteil von PHP - eine große Anzahl von Fragmenten, benutzerdefinierten Funktionen und vorgefertigten Lösungen für häufige Probleme im Internet.

Möglichkeiten von PHP

PHP ist nicht schwer zu verstehen, und es genügt, die Grundlagen zu kennen. Es kann Daten speichern, ändern und löschen. Alles wird innerhalb eines Webservers erledigt (wo es Quellcodes von Webseiten gibt). PHP-Skript wird zuerst auf dem Server ausgeführt und sendet nur das Ergebnis an den Explorer (d.h. es berechnet zuerst 300/30 und sendet dann die Nummer 10 an den Explorer). Daher gibt es im Quellcode nur "10" (im Gegensatz zu JavaScript, das direkt im Explorer berechnet). Im Gegensatz zu JavaScript und HTML wird der PHP-Quellcode im Explorer nicht angezeigt.

PHP kann für die Erstellung eines Diskussionsforums, Gästebuchs, Counters, Meinungsumfrage, Grafik verwendet werden; mit einem einfachen Code ist es möglich, den gesamten Inhalt des Webs zu verwerten. Darüber hinaus ist es möglich, die Seiten mit Datenbanken, z.B. MySQL, zu verbinden.

Zweck von PHP

Es gibt mindestens eine Funktion, die jedes Web nutzen kann. Auf Webseiten werden bestimmte Teile oft wiederholt: Kopf mit Links, Menü, Fußzeile. Mit PHP ist es einfach, eine Vorlage für ein Web zu erstellen, in die die Dateien mit Menü und Fußzeilen eingefügt werden können. Auf diese Weise ist es möglich, das Menü nur einmal zu schreiben und in andere Seiten zu kopieren. Und das Wechseln des Menüs ist sehr einfach (siehe PHP-Menü).

PHP-Dateien

Webseiten mit PHP-Elementen haben meist die Erweiterung.php, aber es kann auch andere Erweiterungen geben, z.B..phtml, php3, php4, php5. Einige Hostings haben bestimmt, welche Version des PHP-Skripts basierend auf der Erweiterung gestartet werden soll (aktuelle Version ist 7). Dies ist jedoch ein Ausnahmefall; meistens genügt .php.

Installation

PHP ist eine Sprache, die nicht nur mit einer bestimmten Version eines Explorers funktioniert (im Gegensatz zu HTML oder JavaScript). Es muss auf dem Computer installiert sein. Grundlage sind ein Webserver und Bibliotheken. Um PHP zu unterstützen, ist es notwendig, es zu installieren und einen Server (meist Apache) zu konfigurieren. Es ist empfehlenswert, PHP Tread für die Installation des PHP-Programms zu verwenden.

Webhosting mit PHP

Nicht jedes Webhosting beinhaltet PHP-Unterstützung. Der Support für Webhosting ist ein überdurchschnittlicher Service gegen Aufpreis. Es ist jedoch möglich, kostenloses Webhosting mit PHP-Unterstützung zu erhalten (z.B. Webzdarma.cz, PHP 5). Bei der Auswahl von Webhosting für PHP-Seiten ist es notwendig, sorgfältig zu lesen, was das Angebot enthält.

PHP - Grundlegende Informationen

Dynamische Seiten (d.h. Seiten, die zuerst von einem Server generiert und dann an den Client gesendet werden) sind heutzutage ein wesentlicher Bestandteil jeder komplexeren Website. Die wichtigsten Skriptsprachen für die Erstellung solcher Seiten sind ASP (Active Server Pages) und PHP. Im folgenden Teil werden wir uns mit der Einführung in PHP beschäftigen.

5.1. Was ist PHP?

PHP ist eine serverseitige Skriptsprache, die in einen gemeinsamen HTML-Code eingefügt wird. Was bedeutet das? Jede Seite, die PHP-Skripte enthält, wird vom Server übernommen und alle Befehle auf der Seite in PHP aufgelistet, dann wird ein sauberer HTML-Code an den Client gesendet (was ein Ergebnis des Skripts ist). Theoretisch kann der Server in allen gesendeten Dateien nach PHP-Skripten suchen, aber meistens ist er so konfiguriert, dass er nur in den Dateien mit den Erweiterungen.php,.php3 oder.phtml nach ihnen sucht. PHP-Befehle werden direkt in den HTML-Code eingefügt und durch die Tags <? und ?> (oder <?php und ?>) getrennt.

Wofür ist PHP gut?

PHP ist eine sehr vielseitige Sprache, in der es relativ einfach ist, z.B. einen Newsserver oder einen virtuellen Shop zu programmieren. Die Daten können in gängigen Textdateien oder in einer Datenbank gespeichert werden (PHP ist mit fast allen gängigen Datenbanken, z.B. mySQL, kompatibel). Die Verarbeitung der Daten aus den Formularen ist sehr einfach, verschiedene einfache Online-Tests einschließlich der Erfolgsquote der Besucher können einfach erstellt werden; auch ein hochwertiges Werbesystem kann einfach programmiert werden. Die Stärke von PHP zeigt sich in der Verwendung auf den Servern Email.cz, Centrum.cz oder BillBoard.cz.

Was ist notwendig, um in PHP zu arbeiten?

PHP wird in HTML eingefügt, so dass jeder gängige HTML-Editor für die Erstellung von PHP-Skripten verwendet werden kann. Es ist sogar genug, um Notepad zu benutzen. Das Wichtigste ist, die erstellten Skripte platzieren und testen zu können. Dazu muss auf dem Server PHP-Unterstützung installiert sein (PHP, jetzt in Version 4, kann kostenlos unter www.php.net heruntergeladen werden). PHP ist als Modul des Linux-Systems am effizientesten, kann aber auch unter Windows verwendet werden (diese Informationen benötigen Sie nur, wenn Sie der Administrator Ihrer Webseite sind). Wenn Sie eines der kostenlosen Webhosting-Angebote nutzen, ist die PHP-Unterstützung nicht sehr wahrscheinlich, aber z.B. der Server www.kgb.cz bietet kostenloses Webhosting mit PHP-Unterstützung (mit einigen Einschränkungen, z.B. ist es nicht möglich, Datenbanken zu verwenden, etc.). Im Falle von Paid Webhosting gibt der Anbieter Auskunft über PHP -

entweder kostenlos oder gegen eine bestimmte Gebühr. Der Anbieter soll auch sagen können, welche Erweiterung für die Dateien mit PHP-Skripten notwendig ist (wie bereits erwähnt, meist sind dies .php,.php3 oder.phtml).

Erstes Skript in PHP

Hier ist das erste PHP-Skript, das die aktuelle Zeit schreibt:

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; char-
        set=windows-1250">
        <title>PHP - Beispiel 1</title>
    </head>

    <body bgcolor="#FFFFFFFF" text="#000000">>
        <center><font face="Arial CE, Arial" size="5">
            Aktuelle Zeit: <?php echo Datum ("H:i:s"); ?>
        </font></center></center>
    </body>
</html>
```

Anweisungen: Speichern Sie im Server z.B. als Datei example1.php und betrachten Sie sie in einem Explorer. Damit ein Skript funktioniert, muss es zuerst von einem Server interpretiert werden, d.h. es konnte nicht offline von einer Festplatte aus untersucht werden.

Wie funktioniert das Skript?

Es ist zu beachten, dass es sich im Grunde genommen um eine klassische HTML-Seite handelt, die auch einen PHP-Befehl enthält - `echo Date ("H:i:s"),` der vom restlichen HTML durch die Tags `<?php` und `?>` getrennt ist. Der Server nimmt zuerst die erforderliche Datei example1.php, und da seine Erweiterung .php ist, lässt er sie durch den PHP-Interpreter laufen und führt alle Befehle aus - diese werden zwischen den Tags `<?php` und `?>` gesucht. Es stößt auf den Befehl echo, dessen Server zum Schreiben in die resultierende Datei dienen. Die Funktion Date gibt das Datum und die Uhrzeit zurück, in den Klammern gibt es die Parameter, die das Format bestimmen, in dem die Uhrzeit angezeigt werden soll (Stunden, Minuten und Sekunden getrennt durch Doppelpunkte - Funktion Date wird später behandelt). Das folgende Semikolon dient zur Trennung der einzelnen Befehle. Es ist in diesem Fall redundant, da es nur einen Befehl gibt, aber es wird empfohlen, sich an das Schreiben zu gewöhnen. Anstelle von PHP-Befehlen

schreibt der PHP-Interpreter die aktuelle Zeit und der Server sendet die Seite dann an den Besucher. Wenn Sie die Quelle des Skriptergebnisses anzeigen möchten, sehen Sie nur ein reines HTML, nicht PHP. Dies führt dazu, dass im Gegensatz zu clientseitigen Sprachen (JavaScript) niemand an Ihren Code herankommt und er daher nicht kopiert werden kann.

Zusammenfassung

- PHP-Befehle werden in einen gemeinsamen HTML-Code eingefügt, sie werden durch die Tags `<?php` und `?>` (oder nur `<?` und `?>`) getrennt.
- Um nur nach PHP-Befehlen zu suchen, muss die Datei die richtige Erweiterung haben - normalerweise .php,.php3 oder.phtml.
- Der Besucher erhält nur einen reinen HTML-Code vom Server (d.h. den Code, nachdem alle PHP-Befehle ausgeführt wurden).

6. Verarbeitung von http-Requests

6.1. Anfragemethoden

GET

Es ist die am weitesten verbreitete Methode. Es dient zum Abrufen eines Objekts (HTML-Datei, Bild,...) vom Server. Die Antwort ist "cacheable". Die GET-Anforderung wird daher von einer Reihe von Überschriften begleitet, in denen sie angegeben ist, wie alt das Dokument ist, ob es geändert wurde usw. Die GET-Anfrage hat in der Regel keinen Body.

POST

Mit dieser Methode ist es möglich, die Informationen vom Benutzer innerhalb eines Körpers an den Server zu liefern (POST wird oft zum Senden größerer Daten aus Webformularen, zum Hochladen von Dateien usw. verwendet).

HEAD

Ist ähnlich wie GET, aber der Körper wird in der Antwort nicht übertragen. Diese Anforderung kann z.B. genutzt werden, um herauszufinden, ob das Objekt wirklich existiert (zur Steuerung der Links innerhalb einer Seite).

PUT/DELETE

Erstellt/löscht das angegebene Objekt vom Server. Diese Methoden werden in der Praxis nicht sehr häufig eingesetzt.
OPTIONS
Es wird verwendet, um Informationen über einen bestimmten Kontext (oder "*" für den gesamten Server) zu erhalten. Der Client kann herausfinden, welche Anfragen er an den jeweiligen Kontext senden kann.

OPTIONEN * HTTP/1.1
Host: www.root.cz

Beispiel für eine implizite Einstellung des Servers

TRACE

Es wird zur Verfolgung des gesamten Anforderungspfades verwendet. Im Body einer Antwort erhält der Client die Anforderungen der einzelnen Systeme, über die die Anforderung ging. Diese Methode wird von Administratoren und Webprogrammen verwendet, die herausfinden möchten, warum der Server ein abgelaufenes Dokument zurückgibt, etc.

Headers

Das Protokoll HTTP Version 1.1 definiert eine große Anzahl von Headern für Anfragen und Antworten. Es gibt einige von ihnen.

6.1.1. REQUEST HEADERS

Accept*

Header dieses Typs geben an, was der Client verarbeiten kann. Der Server wählt dann die am besten geeignete Alternative aus. Zu den Überschriften gehören Accept (MIME-Dokumenttypen, Accept-Charset (Zeichensatz, sehr wichtig in der tschechischen Umgebung), Accept-Encoding (Codierung der übertragenen Daten, wird meist zur Auswahl einer Kompression verwendet) und Accept-Language (Sprache des Dokuments).

Connection

Im Protokoll HTTP 1.1 ist der Parameter "close" definiert, der ein sofortiges Schließen der Verbindung nach der Übertragung des ersten angeforderten Dokuments erfordert.

Referer

Mit diesem Header gibt der Client die URL der Seite bekannt, von der aus der Link generiert wurde.

Host

Mit HTTP 1.1 wird die Unterstützung von sogenannten namensbasierten virtuellen Servern eingeführt. Diese Methode ermöglicht es, mehr virtuelle Server von einer IP-Adresse aus zu betreiben, aber der Client muss den Namen des Servers angeben, mit dem er kommunizieren möchte.

User-Agent

Dieser Header wird verwendet, um das Kundenprogramm zu identifizieren, entweder für statistische Zwecke oder um verschiedenen Forschern unterschiedliche Inhalte zur

Verfügung zu stellen usw.

6.1.2. RESPONSE-HEADER

Content*

Überschriften, die den Inhalt (Körper) der Antwort beschreiben. Es kann z.B. die Länge des Inhalts, seinen MD5-Digest (Content-MD5), die Sprache (Content-Language), den Typ des Dokuments (Content-Type) und andere Attribute enthalten. Es ist zu beachten, dass diese Überschriften nicht nur in den Antworten verwendet werden. Wenn ein Body auch die Anforderung enthält (z.B. bei der POST-Methode), müssen diese ebenfalls verwendet werden.

Server

Dieser Header wird verwendet, um den Server zu identifizieren (normalerweise gibt es seinen Namen, seine Version und manchmal auch andere Informationen).

Verfällt

Ein Server kann diese Informationen verwenden, um den Ablauf eines Dokuments anzuzeigen. Nach dieser Zeit sollte der Client eine neue Version herunterladen.
Es gibt eine Reihe weiterer Header, die z.B. für die Steuerung des Dokumenten-Downloads verwendet werden können ("Download nur, wenn das Dokument seit.... geändert wurde") oder um dem Server einen Benutzernamen und ein Passwort für den Zugriff auf die nicht öffentlichen Teile des Servers zur Verfügung zu stellen. Ebenso kann ein Server seine Antwort genauer beschreiben und den Client informieren, wann das Dokument zum letzten Mal geändert wurde oder ob sein Caching in öffentlichen oder privaten Caches erlaubt ist.

6.2. Grundlegende Merkmale des HTTP-Protokolls

Um die Informationen vollständig zu verstehen, ist es notwendig, die grundlegenden Merkmale des HTTP-Protokolls und seine Funktionsweise zu kennen. Das HTTP-Protokoll ist ein Protokoll auf Anwendungsebene für verteilte hypermediale Informationssysteme. In der Praxis bedeutet dies, dass dieses Protokoll im Allgemeinen im Internet nicht nur zur Datenübertragung zwischen einem Client und einem Server, sondern auch für viele andere Zwecke verwendet wird. Das HTTP-Protokoll ist zustandslos, d.h. es erkennt nicht die Clients, von denen Anfragen gestellt werden. Wenn ein Client eine Anfrage sendet und dann eine andere sendet, erkennt der Server nicht, dass es sich um den gleichen Client handelt.

HTTP existiert in 3 Versionen - 0.9, 1.0 und 1.1 . Das erste von ihnen, das als HTTP/0.9 bezeichnet wird, existierte als einfaches Protokoll, das Daten im Internet nur eingeschränkt übertragen konnte. Die Version HTTP/1.0 ermöglichte die Übertragung von Daten im MIME-Format, so dass sie auch Metainformationen über die übertragenen Daten enthalten konnte. Die wichtigste Verbesserung gegenüber der Version HTTP/1.1, die auch die neueste Version ist, war, dass alle Verbindungen dauerhaft wurden, was bedeutet, dass die Verbindung geschlossen wird, wenn entweder der Client oder der Server einen Header zum Schließen sendet. HTTP verwendet, um die Verbindung nach jeder Antwort vom Server zu schließen. Diese Verbesserung beschleunigte die Übertragung erheblich, da der Server nicht für jedes Bild, jeden Frame und jedes Applet eine neue Verbindung öffnen muss.

6.2.1. HTTP-PROTOKOLL ANFORDERUNGSFORMAT

Die Anforderung des HTTP-Protokolls hat folgendes Format:

METHODE URL DES HTTP-DOKUMENTS
HEADER
Leerzeile
ANDERE DATEN nur bei der Methode POST

Die Anforderungsmethode gibt an, wie der Server die Anforderung verarbeiten soll. Die Methoden werden später behandelt. Header werden im folgenden Format gesendet:

NAME DES HEADERS: HEADER-WERT

Jeder Header muss in einer separaten Zeile stehen. Alle Zeilen müssen mit den Tags CRLF (\r\n) abgeschlossen werden. Am Ende aller Überschriften muss eine Leerzeile stehen, auch wenn keine weiteren Daten vorhanden sind.
Anfragen in PHP werden über sogenannte Sockets gesendet. Ein Socket ist im Grunde genommen eine Verbindung zwischen einem Client und einem Server. Um mit ihnen zu arbeiten, ist es notwendig, zuerst das Socket zu öffnen. Zum Öffnen wird die Funktion *fsockopen* verwendet:

```
fsockopen(server, port);
```

Beispiel:

```
$sock = fsockopen("www.interval.cz", 80);
```

Bei geöffneter Socket ist es möglich, die Anforderung über die Funktion *fputs* zu senden:

```
fputs(socket, request);
```

Beispiel:

```
fputs($sock, "GET /index.html HTTP/1.1\r\nHost:
www.interval.cz\r\n\r\r\n")
```

Im folgenden Teil werden wir uns mit HTTP-Methoden befassen.

7. HTTP-Protokoll-Anforderungsmethoden

In HTTP/1.1 gibt es sieben grundlegende Methoden von HTTP-Requests:

- GET
- POST
- HEAD
- OPTIONS
- PUT
- DELETE
- TRACE

Nach jeder Anfrage können einzelne Header folgen. In der Version HTTP 1.1 ist es in jeder Anfrage zwingend erforderlich, den Header Host zu verwenden, der den Host angibt. Nach den Überschriften muss eine Leerzeile folgen, wie bereits erwähnt.

7.1. GET-Methode

Die GET-Methode ist die einfachste und eine der grundlegenden Methoden. Jedes Mal, wenn eine Seite von einem Server geladen wird, ohne zuerst ein Formular mit der POST-Methode zu senden, wird diese Methode verwendet, um die Seite vom Server abzurufen. Das Ergebnis ist also eine Seite und ihre Überschriften, nach denen wir bei dieser Methode fragen. Das Format dieser Methode ist wie folgt:

LIEFERT URL DER SEITE DER PROTOKOLLVERSION
HEADERS
Leerzeile

Beispiel:

GET /index.asp HTTP/1.1.1
Host: www.interval.cz
Leerzeile

7.2. POST-Methode

Die POST-Methode arbeitet wie die GET-Methode, aber im Falle der POST-Methode ist es möglich, Daten an das Skript nach den Überschriften und der Leerzeile zu senden. Diese Methode wird zum Senden der Daten aus einem Formular mit der POST-Methode verwendet. Das Format dieser Anfrage ist wie folgt:

POST URL- DER SEITE DER PROTOKOLLVERSION
HEADERS
Leerzeile
DATEN AUS DEM FORMULAR

Beispiel:

POST /processdata.php HTTP/1.1
Host: www.formulare.cz
Inhaltslänge: 29
Content-Type: application/x-wwww-form-urlencodiert
Leerzeile

array1=value1&array2=value2=value2

Bei dieser Methode gibt es Mroe Header, die noch nicht erwähnt wurden. Content-Length gibt die Länge der Daten aus dem Formular (in Bytes) an, und Content-Type: application/x-www-form-urlencoded bezeichnet den MIME-Typ der Daten aus dem Formular.

HEAD, OPTIONEN, PUT, DELETE, TRACE

Wenn Sie keinen Internet Explorer programmieren, werden Sie wahrscheinlich nicht auf diese Methoden stoßen. Die folgende Tabelle zeigt daher nur die grundlegende Bedeutung dieser Methoden.

HEAD	Es funktioniert als GET, aber es gibt nicht den Body der Seite zurück, sondern nur den Header. Dies wird z.B. verwendet, um festzustellen, ob sich die Seite gegenüber dem letzten Aufruf geändert hat.
OPTIONS	Es wird für Anfragen nach Servermöglichkeiten verwendet.
PUT	Es wird als GET bezeichnet, behält aber den Körper der Anforderung an einer Stelle, die durch die erforderliche URL vorgegeben ist. Es ist vergleichbar mit dem Senden von Dateien per FTP.
DELETE	Es entfernt das Dokument vom Server. Das zu entfernende Dokument erhält die URL der Anfrage.
TRACE	Es wird verwendet, um die Anfrage über alle Proxy-Server und Firewalls zu verfolgen, die die Anfrage durchläuft. Es ist vergleichbar mit dem Tool TraceRoute.

7.3. Verwendung von Datenquellen

Moderne und effiziente Datenverarbeitungsanwendungen speichern Informationen nicht direkt in ihren eigenen Dateien, sondern nutzen einige der externen Datenquellen. Es gibt eine Reihe von Datenquellen, angefangen von einer einfachen Textdatei über Dateidatenbanken z.B. im DBF-Format bis hin zu Datenbanken, die auf SQL-Servern gespeichert sind.
Jede dieser Quellen verwendet in der Regel ein eigenes Format für die Speicherung von Daten und Methoden für den Abruf der gespeicherten Daten. Um das Problem mit der Vielfalt der Datenquellen zu vermeiden und die Notwendigkeit zu vermeiden, Anwendungen für jede dieser Quellen zu entwickeln, haben die Anwendungsprogrammierer ein Standardwerkzeug entwickelt, das den Zugriff auf verschiedene Datenquellen über eine standardisierte Plattform - ODBC - ermöglicht.

Eine solche standardisierte Plattform ermöglicht es den Nutzern, die Daten aus verschiedenen Quellen an verschiedenen Standorten komplex und einfach zu nutzen. So ist es beispielsweise möglich, die Daten aus Access in Word oder Excel zu verwenden, ohne sie exportieren und zuerst in die gewünschte Anwendung laden zu müssen.

7.3.1. WAS IST ODBC?

ODBC steht für Open Database Connectivity. ODBC-Datenquellen sind für Anwendungen über einen relevanten Treiber zugänglich, der als Vermittler für die Kommunikation zwischen einer user´s Anwendung und einer externen Datenquelle wahrgenommen werden kann. Die Anwendung sendet die Anfrage nach den Daten an ODBC. Der jeweilige Fahrer übersetzt die Anforderung so, dass die externe Datenquelle sie versteht und an die Datenquelle sendet. Die Antwort der Datenquelle wird ebenfalls über ODBC gesendet, das das Ergebnis in das Standardformular übersetzt und an die Anwendung zurückgibt.

Das Prinzip von ODBC ist Standard; daher kann jede Anwendung, die den ODBC-Treiber verwenden kann, auf jede externe Datenquelle von jedem Hersteller zugreifen, der den entsprechenden Treiber bereitstellt. ODBC ermöglichte es so, den Zugriff auf die Daten für die Anwendungen zu standardisieren, ohne zu lösen, wie die Datenquelle funktioniert.

Einerseits muss die Anwendung nur wissen, wie man mit ODBC arbeitet. Andererseits stellen die Hersteller verschiedener Datenquellen die relevanten ODBC-Treiber zur Verfügung, so dass ihre Quellen für die Anwendungen leicht zugänglich sind. Dies ist sowohl für die Anwendungsprogrammierer, die einen unabhängigen, standardisierten Zugriff auf Daten haben, als auch für die Benutzer, die die Anwendungen erhalten, die kostengünstiger und schneller auf verschiedene Daten zugreifen können.

Arbeiten mit ODBC unter Windows

ODBC-Setup-Steuerelemente sind bereits ein integraler Bestandteil des Betriebssystems Microsoft Windows. Der ODBC-Datenquellenadministrator befindet sich im Bedienfeld. Dabei ist zu beachten, dass die Verfahren und Bilder im Betriebssystem Windows 10 gültig sind, aber auch in anderen Versionen von Windows. In einigen Betriebssystemen kann der Zugriff auf den ODBC-Administrator jedoch auf eine andere Weise erfolgen.

8. Datenquellen

Um die Mailinglisten und Vorlagen zur richtigen Zeit mit den richtigen Daten zu füllen, verwendet Mailkit XML- und RSS-Datenquellen. Ihre Nutzung ist je nach Art des Benutzerkontos begrenzt. Die Version Mailkit Base ist auf die Verwendung von XML-Datenquellen beschränkt, die nur für den Import der Empfänger in Listen bestimmt sind, während Mailkit Syndicate und Agency sowohl XML- als auch RSS-Datenquellen unterstützen, auch für das Laden von Daten in Vorlagen.

Verwendung von XML-Datenquellen für die Empfängerliste

Um eine neue Empfängerliste aus einer Datenquelle zu erstellen, ist es notwendig, eine neue XML-Datenquelle einzurichten.

- Name - Name der Datenquelle. Wenn die Datenquelle für die Empfängerlisten funktioniert, hat die Liste den gleichen Namen.
- Beschreibung - Beschreibung der Datenquelle.
- Quelle - URL-Adresse des XML oder RSS, das als Datenquelle dient.
- Autorisierung - es ist angekreuzt, wenn der Zugriff auf den Speicherort der Datenquelle passwortgeschützt ist.
- Typ - Sie können zwischen RSS- oder XML-Datenquelle wählen.
- Ziel - das Ziel der Datenquelle wird aus der Auswahlliste ausgewählt. Es wird entweder für die Liste der Empfänger oder für die Vorlage verwendet.
- Automatisches Update - wenn diese Option aktiviert ist, wird die Quelle automatisch aktualisiert, bevor die Kampagne gesendet wird.
- Verfallszeit - legt die Zeit fest, nach der die Datenquelle im Falle einer automatischen Aktualisierung aktualisiert werden soll.
- Letztes Update - es zeigt das Datum des letzten Updates der Datenquelle an.
- Leere Datensätze:
- Zurücksetzen - die Empfängerdaten werden entsprechend den leeren Datensätzen im Import gelöscht.
- Aktuellen Wert beibehalten - die Empfängerdaten bleiben in der gleichen Form wie vor dem Import.

Wie man die Datenquelle vorbereitet

Die Datenquelle hat keine definierte Struktur und kann in den Formaten XML oder JSON vorliegen, die voll gültig sein müssen. Die Datenquelldatei muss an eine URL gesendet werden, die von den Mailkit-Servern verfügbar ist, und gegen den Zugriff Dritter geschützt sein, da es sich um sensible Informationen handelt.

Da jeder Kunde ein anderes Informationssystem mit unterschiedlichen Möglichkeiten verwendet, ist das Datenquellen-System universell und setzt die Struktur der benötigten Daten nicht spezifisch fest. Es gibt jedoch einige technische Einschränkungen der Da-

tenquellen:

- Vollständig valides XML oder JSON
- Attribute werden im XML-Format nicht unterstützt (z.B. `Vorname=”Jana” gender=”f” country=”cz”).`
- UTF8-Zeichenkodierung empfohlen
- Verwenden Sie "," nicht, um mehr Werte zu trennen - verwenden Sie "|".
- Der eindeutige Datensatzbezeichner und das einzige obligatorische Feld ist die E-Mail-Adresse. Wenn es mehr Datensätze mit derselben E-Mail-Adresse gibt, werden diese überschrieben.

Wie bereits erwähnt, sind nur E-Mails obligatorisch; andere Daten sind nicht obligatorisch, aber wichtig. Die allgemeine Regel ist "je mehr, desto besser", aber auch nichts darf übertrieben werden. Die Datenquelle ist wichtig für das Maximum der verfügbaren Empfängerdaten, die für bestehende und zukünftige E-Mail-Kampagnen verwendet werden können.

Import von Daten aus der Datenquelle

Um Inhalte zu erhalten, müssen die Quellzweige den Kontaktfeldern zugeordnet werden. Um eine Zuordnung vorzunehmen, klicken Sie auf den Namen der Datenquelle und klicken Sie auf die Schaltfläche View Structure. Nachdem Sie alle Felder zugewiesen haben, klicken Sie auf Speichern. Anschließend ist es möglich, mit Import für den aktuellen Datenimport fortzufahren. In diesem Moment wird eine neue Empfängerliste erstellt (sie hat den gleichen Namen wie die Quelle, die sie erstellt hat), und die Daten aus der Datenquelle werden entsprechend der vorherigen Zuordnung importiert.
Gleichzeitig haben die Benutzer des Syndikats- und Agenturkontos die Möglichkeit der automatischen Aktualisierung. Wenn die automatische Aktualisierung gewählt wird, wird die Datenquelle automatisch importiert und die Empfängerliste wird vor Beginn jeder Kampagnenauslieferung aktualisiert. Dieser Parameter wird für Datenquellen mit mehr als tausend Datensätzen nicht empfohlen, da die Aktualisierung einige Minuten dauern kann, was die Auslieferung der Kampagne verzögert. Für größere Datenquellen wird empfohlen, die Möglichkeit der regelmäßigen geplanten Aktualisierung durch den Kundensupport für eine bestimmte Tageszeit zu nutzen.

Verwendung von XML- und RSS-Datenquellen in Vorlagen

Die Einrichtung von XML- und RSS-Datenquellen für die Verwendung in Vorlagen ähnelt der Verwendung für Empfängerlisten, ohne jedoch jedem Feld die Bedeutung zuweisen zu müssen. Die Werte werden durch eine Reihe von Namen in der Vorlage bestimmt, daher ist es einfach, jede XML- oder RSS-Quelle einzurichten.

Oben gibt es ein Beispiel für einen Template Code, für den die RSS-Datenquelle EXAMPLE verwendet wird. Der Befehl FOREACH erstellt eine Schleife zum Parsen und Suchen aller Datensätze. Jeder der Standard-RSS-Tags ist einfach zu lösen und in den HTML-

Code einzufügen, so dass die Daten in der Vorlage ausgegeben werden können. Weitere Informationen finden Sie unter E-Mail-Vorlagen.

Produktdatenquellen

Datenquellen können auch für die Übermittlung des Produktangebots an Mailkit und für die spätere Verwendung von Produktinformationen in Kampagnen genutzt werden. Hier zeigt sich die Leistungsfähigkeit von Datenquellen und programmierbaren Vorlagen, die es ermöglichen, Daten aus mehreren Quellen zu kombinieren und die Inhalte für die einzelnen Empfänger automatisch zu personalisieren.

Für Produktinformationen ist es möglich, eines der gängigen Produkt-Feed-Formate für Heureka, Zbozi, Google, etc. zu verwenden oder den eigenen Feed mit den notwendigen Informationen zu generieren. Da die Produktfeeds sehr umfangreich sind und die Geschwindigkeit der Arbeit mit den enthaltenen Daten wichtig ist, werden diese Datenquellen direkt in die SQL-Datenbanken übertragen, und es ist weiterhin möglich, mit ihnen zu arbeiten. Für die Einstellung der Produktdatenquelle wenden Sie sich bitte an den Kundensupport, der Ihnen bei der Implementierung und Nutzung hilft.

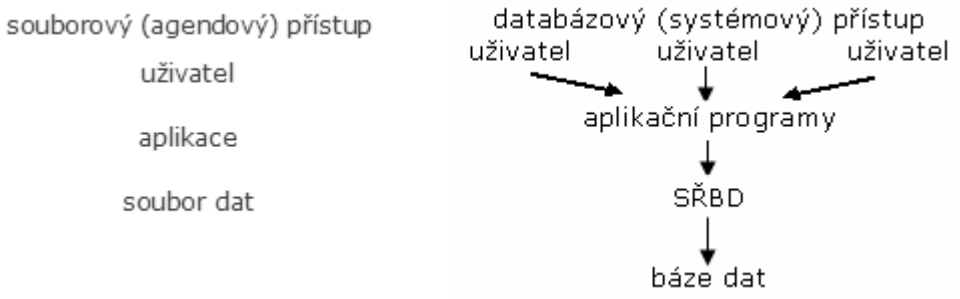
Zustell-Feeds

Delivery Feeds sind spezielle Datenquellen, die zur Weitergabe strukturierter Informationen für die Realisierung der Kampagnenauslieferung verwendet werden. Während eine Kampagne in der Regel die voreingestellte Empfängerliste verwendet, die für die Auslieferung der Kampagne nach den festgelegten Regeln verwendet wird, wird die Kampagne im Falle des Lieferfeed nur an die im Feed aufgeführten Adressen gesendet. Es ist eine Alternative zum API-Aufruf mailkit.sendmail_mass, d.h. der Weg, um beim Versenden der Kampagne an Mailkit hochstrukturierte Daten zu verarbeiten, z.B. aus Personalisierungssystemen oder CRM. Diese Feeds müssen eine streng definierte Struktur im XML-Format haben.

9. Datenbasis Ansatz

Anforderungen an das Datenbanksystem:

- Überprüfung der Datenkonsistenz - Die Datenbank muss in der Lage sein, die Einhaltung bestimmter Regeln der so genannten Integritätsbeschränkungen zu gewährleisten und die Daten vor möglichen Unfällen, die bei Transaktionen auftreten können, zu schützen.
- Transaktion - Reihenfolge der Manipulation mit Daten, die stattfinden muss, um sicherzustellen, dass die Daten ordnungsgemäß gespeichert werden, z.B. Übertragung von einem Konto auf ein anderes Konto (dies muss auf beiden Konten korrekt erfolgen).
- Große Datenmengen - im Vergleich zu den Möglichkeiten von Speichermedien muss die Datenbank in der Lage sein, ausreichende Datenmengen zu speichern.
- Datenmanagement – Entwicklungsstufen



Legende: datová hierarchie - Datenhierarchie, databáze - Datenbank, soubor - Datei, věta, Datensatz - Anweisung, Datensatz, Attribut, Attribut, Pol - Attribut, Array

Design einer strukturierten Datenbank

- Die Realität, deren Reflexion die entworfene Datenbank sein soll. Es besteht aus mehreren Objekten (Entitäten).
- Es können unterschiedliche Beziehungen zwischen den überwachten Einheiten bestehen (z.B. zwischen den Einheiten des gleichen Typs = rekursive Beziehung).

Kardinalität der Beziehung: symbolische Bezeichnung 1:1 (Professor XY´s Frau ist ZN); 1:N (Professor XY Lehrer Schüler); M:N (welche Schüler von welchen Professoren unterrichtet werden)

Integritätsbeschränkungen der Datenbank - alle Regeln, die die zulässigen Werte (eine Kombination von Werten) der Attribute definieren, Anzeigeformat

Relationales Datenmodell

Es wird davon ausgegangen, dass es einwertige Attribute gibt.

- Darstellung in Form einer relationalen Tabelle, in der ein Tupel einer Zeile entspricht und ein Attribut einer Spalte entspricht.
- Relationale Datenbank
- alle Daten liegen in Form von 1 oder mehreren Tabellen vor, die nach Spalten benannt sind.
- jede Spalte enthält Daten aus 1 Domäne (d.h. 1 Datentyp)
- die Elemente der einzelnen Spalten (mit Name und Typ) werden in der Regel als Elemente oder Arrays bezeichnet, und die Begriffszeile entspricht dem Datensatz (Statement).
- Im Sinne eines relationalen Modells beschreiben Sitzungen beide Einheiten und die Beziehungen zwischen ihnen.
- Basierend auf dieser dualen Nutzung ist es möglich, zwischen sogenannten Entity-Sessions (Sätze von arrangierten Tupelattributen, die die Entitäten beschreiben) und Relationship-Sessions (Sätze von arrangierten Tupeln) zu unterscheiden.
- Data Warehouses
- Warehousesprinzip - 2 Hauptziele
- Die Vereinheitlichung der Datenansicht in den einzelnen sogenannten Produktionssystemen bietet einen klaren Zugriff auf die Daten - unterschiedlich aufgerufen, aber die gleichen Objekte werden als ein Objekt betrachtet.
- unterschiedlich gemessen, aber die gleichen Variablen werden mit dem gleichen Messgerät gemessen.

10. Strukturierte und unstrukturierte Daten

10.1. Strukturierte Daten

Grundtypen (Klassifizierung zur Unterscheidung von erlaubten und unzulässigen Manipulationen und Werten)

- Text (Zeichenketten, die Informationen mittels eines Textcodes ausdrücken, bestimmte Mengen von Elementen, die aufgezeichnet werden können oder die eine Syntax definieren können)
- numerisch - reale rationale Zahlen
- Datum, Uhrzeit - Grenzwerte, die es erreichen kann (30. Februar, 27:00 Uhr)
- logisch - erfüllt die Bedingungen für das Vorhandensein oder Nichtvorhandensein eines Objekteigenschaften - 2 Werte (0 und 1, A und N)
- Kategorie - Wert der Merkmale, die aus einer Skala ausgewählt werden (oft wählt man, ermöglicht die Aufzeichnung der Werte nur mit Hilfe eines Codes).
- Strukturierung schafft eine Datenorganisation, die eine effiziente Speicherung, Verarbeitung und Abruf von Daten nach Bedarf ermöglicht → Strukturierte Daten erstellen Suchschlüssel (manchmal auch als Identifikationsschlüssel bezeichnet) - Schlüssel, die Datensätze eindeutig identifizieren, werden als private Schlüssel (Identifikationsschlüssel) bezeichnet - eine Grundvoraussetzung für Daten und Datenbanksystem.
- Informationen über etwas (Widerspiegelung der Realität) - Vor- und Nachname, Adresse, Alter, Telefonnummer, Gewicht, Preis,.... Punkteanzahl, Kategorie, Durchschnittsnote,... Stückzahl, Seitenzahl
- Operationen oder was getan werden kann - Addition, Rundung, Multiplikation, Verbindung (Name+Nachname), Verkürzung, Sortierung,...., Tag in einer Woche, Negation, ...
- Datentyp (muss definiert werden) - Anzahl, Text, Datum und Uhrzeit, logische Daten (ja/nein)
- Verschlüsselte Daten - verschiedene Kodierungen - Text, Buchstaben - verschiedene Codetabellen (ascii, ebdic,....) nationales Alphabet; Datum und Uhrzeit (wie wir ein Datum schreiben)

10.2. Unstrukturierte Daten

Datentyp: Freitext, Audio, Video, Grafik, Multimedia, etc.

- Bieten Sie mehr Informationen als strukturierte Daten.
- Problem: Es ist sehr schwierig, mit unstrukturierten Daten zu suchen. Lösung: Unstrukturierte Daten werden durch strukturierte Daten ergänzt (mp3-Titel).

Datenmengen - strukturierte und unstrukturierte Daten

- ascii Textseite (Notepad) 1,8 kB
- Worttext Seite 50 kB
- Vektorgrafik A4 30 kB
- Bitmap-Bild A4(jpg, rgb) 5 MB
- 1-minütige Audioaufnahme (WAV) 10 MB
- 90-minütige Videoaufzeichnung 3 GB

Data Warehouse (DWH) ist eine spezielle Form der relationalen Datenbank, die es ermöglicht, Aufgaben zu lösen, die sich hauptsächlich auf die analytische Abfrage über große Datensätze konzentrieren.

Fachliche Orientierung

Im Falle einer gemeinsamen relationalen Datenbank ist es üblich, die Redundanz der gespeicherten Daten zu minimieren, die durch ihre Normalisierung in die dritte Normalform und die interne Verknüpfung der einzelnen logischen Funktionseinheiten erreicht wird. Im Data Warehouse wird immer auf eine klare interne Trennung der einzelnen Funktionseinheiten geachtet. Das Ergebnis ist eine klarere Struktur für einen Benutzer (Manager, Business Analyst) auf Kosten der erhöhten Anforderungen an einen Speicherplatz.

Integration

Eine gemeinsame operative Anwendung (ein Programm) über eine relationale Datenbank löst eine bestimmte spezifische Art von Aufgaben über "ihre" spezifischen Daten. In einem Data Warehouse ist es notwendig, die Informationen aus vielen verschiedenen Quellen zu sammeln und nicht nach ihrer Herkunft, sondern nach ihrer logischen Bedeutung zu gruppieren (sie sind eng mit der Themenorientierung verbunden - alle Daten zu einem bestimmten Funktionsbereich müssen "auf einem Stapel" liegen, unabhängig davon, woher sie kommen).

Geringe Variabilität

Die Daten werden in der Regel in größeren Chargen (z.B. in Wochen- oder Tagesintervallen) in ein Lager hochgeladen und nicht nachträglich verändert.

Historisierung

Die Daten in einem Data Warehouse werden in der Regel in ihrer historischen Form gespeichert, nicht nur im aktuellen Zustand. Dies ist auf die Notwendigkeit zurückzuführen, Analysen durchzuführen, die sich auf die Entwicklung im Zeitablauf konzentrieren. Aus der Sicht von users´ ist in einer gemeinsamen relationalen Datenbank nur der aktuelle Zustand der Datenobjekte interessant.

10.2.1. TECHNOLOGISCHE MERKMALE VON DATA WAREHOUSES

Die Anforderungen an Data Warehouses implizieren ihre technologischen Eigenschaften:

- Ein Data Warehouse muss ein Werkzeug zum Hochladen von Daten aus verschiedenen Datenquellen enthalten. Diese Quellen können unterschiedliche Datenformate und unterschiedliche physische Standorte haben. Es müssen nicht nur relationale Datenbanken sein.

- Ein Data Warehouse speichert Daten nicht unter Berücksichtigung der besten Bearbeitungsbedingungen, sondern im Hinblick auf die beste und schnellste Ausführung komplexer Anfragen. Daher wird für die Datenspeicherung häufig die OLAP-Technologie eingesetzt.

- Es kann nicht gesagt werden, welche Anforderungen und Aufgaben die Benutzer in Zukunft haben werden (zum Zeitpunkt der Erstellung eines Data Warehouse kennen wir nur die Art der Aufgaben, nicht die einzelnen Aufgaben und Anforderungen).

10.2.2. LOGISCHE STRUKTUR DES DATA WAREHOUSE

Aus logischer Sicht (user´s) sind die Daten in einem Data Warehouse in Diagramme unterteilt. Jedes Diagramm entspricht einem analysierten Funktionsbereich.
Der Kern jedes Diagramms besteht aus einer oder mehreren Faktentabellen. Sie speichern die analysierten Daten - numerische und finanzielle Werte, die für analytische Berechnungen verwendet werden - Aggregation, Sortierung, etc. Der meiste Speicherplatz in einem Data Warehouse wird durch Faktentabellen belegt, die detaillierte Informationen aus allen Quellen enthalten, d.h. mehr als andere Tabellen.

Faktentabellen werden über Fremdschlüssel mit Dimensionen verknüpft. Dimensionen

sind Tabellen, die Listen von Werten enthalten, die zur Kategorisierung und Sortierung von Daten in Faktentabellen verwendet werden.

Beispiel

Es ist notwendig, Informationen über alle Verkäufe aus den Registrierkassen von Hypermärkten in einem Data Warehouse zu speichern. Die Daten werden anhand des Verkaufszeitpunkts, des Geschäfts, der Warenart, des Lieferanten, der laufenden Marketingveranstaltungen und der Zahlungsmethode (mit Karte, in bar) weiter analysiert.

Das Verkaufsdiagramm enthält eine Faktentabelle - Verkaufsartikel, in der Informationen über die Art der verkauften Ware, den Preis und die Stückzahl (oder die verkaufte Menge) für jeden verkauften Artikel gespeichert werden.

Neben dieser Faktentabelle enthält das Diagramm auch Dimensionen für die Sortierung der Verkaufsartikel: Zeitdimensionen Datum und Stunde (innerhalb eines Tages), Dimensionen Shop, Warenart, bei der es für jeden Artikel eine Zeile gibt (z.B. Erdbeerjoghurt 250 ml), Warengruppe (Joghurt), Abteilung (Milchprodukte), Lieferant (Molkerei XX), etc.

Die Faktentabelle muss mit jeder dieser Dimensionen über einen Fremdschlüssel direkt oder indirekt verknüpft sein.

Es gibt zwei Möglichkeiten, hierarchische Dimensionen zu speichern:

- Aus der gesamten Hierarchie wird eine Dimensionstabelle erstellt, die die Daten für die höheren Hierarchieebenen redundant speichert. Auf diese Weise wird ein Diagramm erstellt, bei dem jede Dimensionstabelle direkt mit einer Faktentabelle verknüpft ist. Je nach Form wird dieses Diagramm als Sternschema bezeichnet.

- 3NF wird auf die hierarchische Dimension angewendet, so dass nur die Dimension auf der untersten Ebene der Hierarchie direkt mit der Faktentabelle verknüpft wird. Andere Dimensionen werden mit jeder der niedrigeren Dimensionen in der hierarchischen Struktur verknüpft. Dies wird als SnowFlake-Schema bezeichnet.

11. AJAX

AJAX steht für Asynchronous JavaScript and XML. Es ist ein Oberbegriff für Technologien zur Entwicklung interaktiver Webanwendungen, die den Inhalt ihrer Seiten verändern, ohne sie vollständig hochladen zu müssen. Sie erfolgt durch asynchrone Verarbeitung von Webseiten mittels einer Bibliothek in JavaScript. Im Gegensatz zu klassischen Webanwendungen sind sie benutzerfreundlicher, erfordern aber die Verwendung moderner Webbrowser.

Diese Anwendungen werden mit den folgenden Technologien entwickelt:

- HTML (oder XHTML) und CSS zur Darstellung von Informationen;
- DOM und JavaScript zur Anzeige und dynamischen Änderung der präsentierten Informationen;
- XMLHttpRequest für den asynchronen Datenaustausch mit einem Webserver (typischerweise wird das XML-Format verwendet, aber es ist auch möglich, andere Formate wie HTML, Text, JSON oder EBML zu verwenden).
- Wie DHTML, LAMP oder SPA ist Ajax keine konkrete, unabhängige Technologie. Es ist ein Begriff, der sich auf mehrere Technologien und ein bestimmtes Ziel bezieht.

Vorteile

Zu den Vorteilen gehört der Wegfall der Notwendigkeit, die gesamte Seite in jedem Vorgang neu zu laden und zu zeichnen (im Gegensatz zum klassischen Modell der WWW-Seiten). Wenn ein Benutzer in einer Meinungsumfrage auf einen Vote-Button klickt, muss die gesamte Seite vom Server neu geladen werden, auch wenn nur die Vote-Ergebnisse aktualisiert werden und der Rest des Inhalts gleich bleibt. Mit AJAX wird die Abstimmung im Hintergrund gesendet, der Server sendet nur die Teile der Seiten, die sich geändert haben, und diese Teile werden aktualisiert und auf der Seite neu gezeichnet. Außerdem gibt es keinen unangenehmen Effekt, wenn nach der Aktivität ihre Blockelemente, Bilder usw. auf der kontinuierlich hochgeladenen Seite schrittweise angepasst und formatiert werden. Es kann auch ärgerlich sein, dass nach der spezifischen Aktivität, in der Mitte einer längeren Seite (nach unten gescrollt), die neu hochgeladene Seite nach oben gescrollt angezeigt wird. Mit AJAX arbeitet der Benutzer flüssiger und die Geschwindigkeit (insbesondere bei einer schnelleren Internetverbindung) ist fast gleich der Geschwindigkeit von gängigen Desktop-Anwendungen.

Dies hat auch das Potenzial, die Belastung der Webserver und des Web im Allgemeinen zu reduzieren. Da nicht für jeden Request das gesamte HTML-Dokument erstellt werden muss, sondern nur die vorgenommenen Änderungen markiert werden müssen, ist das Datenvolumen deutlich geringer und kann theoretisch auch die Auslastung der Datenbankserver oder anderer Backend-Systeme positiv beeinflussen.

Nachteile

Andererseits kann AJAX die Anzahl der ausgetauschten HTTP-Requests erhöhen, und obwohl sie ein geringeres Datenvolumen übertragen, sinkt die Last bei unsachgemäßer Implementierung nicht.

Zu den Nachteilen gehören vor allem die Änderungen im Paradigma der Webnutzung: Webseiten verhalten sich wie Anwendungen mit einer komplexen internen Logik, nicht wie eine Folge von Seiten, die auch mit den Schaltflächen Zurück und Weiter navigiert werden können. Ebenso ist es nicht möglich, die URL der Seite, auf der etwas "geklickt" wurde, mit der AJAX-Technologie zu übergeben. Moderne AJAX-Anwendungen sind in der Lage, das Browsen im Verlauf (zumindest teilweise) mit verschiedenen Techniken wiederherzustellen (z.B. mit einem Teil der Adresse nach dem # oder mit unsichtbaren IFRAMEs). Dies macht jedoch das Design der Seiten komplizierter und zeitaufwendiger, sowie den Aufwand für die Implementierung mit AJAX.

Ein weiteres Problem von AJAX-Anwendungen kann die Netzwerklatenz sein: Der Bedarf an Internetkommunikation hat negative Auswirkungen auf die Antwortgeschwindigkeit und die Interaktivität der Benutzeroberfläche. Wenn der Benutzer nicht deutlich angekündigt wird, dass die Anwendung seine Anfrage bearbeitet (und mit dem Server im Hintergrund kommuniziert), registrieren sie lediglich eine verzögerte Antwort (die Benutzer können sogar versuchen, den Vorgang erneut zu starten, da sie denken, dass das System die Anfrage ignoriert hat, wodurch eine höhere Last für den Server erzeugt wird und / oder etwas verursacht wird, das sie nicht geplant haben, z.B. zehn Tickets statt zwei). Als geeignete Lösung empfiehlt es sich, irgendwie zu zeigen, dass die Anfrage user´s bearbeitet wird, z.B. mit Text oder animiertem Symbol.

Ein weiterer Nachteil von AJAX ist die Notwendigkeit, einen modernen Grafikbrowser zu verwenden, der die notwendigen Technologien unterstützt. Allerdings unterstützen alle derzeit verwendeten gängigen Browser diese Technologien zumindest grundsätzlich; das Problem liegt nur bei den Minderheitsbrowsern von Lynx oder bei hardwareschwachen Browsern, z.B. bei einigen Handys und PDAs. Im Rahmen der Web-Zugänglichkeit ist es erforderlich, dass die Seiten auch ohne AJAX von Browsern aus zugänglich sind, was mehr Zeit und Arbeit für die Entwickler und höhere Kosten für die Käufer der Website bedeutet.

AJAX

Wie bereits erwähnt, steht AJAX für Asynchronous JavaScript and XML. AJAX ist eine moderne Technologie, die in den aktuellen Webanwendungen häufig verwendet wird. Es wird oft erwähnt, da es ein Teil von RIA ist, einem neuen Programmierstil, der zu höherem user´s Komfort und Funktionalität der Anwendungen führt.

AJAX ist keine neue Technologie, es ist nur eine Kombination aus bereits bekannten

Technologien - HTML (oder XHTML), JavaScript, XML und XMLHttpRequest.
Warum ist AJAX so vorteilhaft? Die Anwendungen, die AJAX verwenden, können Daten von einem Server senden und abrufen, ohne die gesamte Seite neu laden zu müssen (im Gegensatz zu klassischen Links). AJAX kann daher für verschiedene Zwecke verwendet werden, z.B. für Autovervollständigungen (Formulare, die je nach gedrückter Taste automatisch ausgefüllt werden), AJAX Meinungsumfragen und andere komplexere Anwendungen, die die Arbeit eines Benutzers erleichtern können.

AJAX hat auch einige Nachteile, insbesondere bei unsachgemäßer Verwendung, es reduziert die Benutzerfreundlichkeit der Seiten erheblich. Wie bei anderen Technologien ist es daher notwendig, die AJAX-Anwendung sorgfältig zu planen und am Anwender zu testen.

Wie Sie bereits wissen, ist Javascript eine so genannte clientseitige Sprache, d.h. es wird auf der Seite des Webbrowsers client´s durchgeführt und vom Javascript-Interpreter interpretiert. Es hat eine Vielzahl von Eigenschaften, die auch die Fähigkeit beinhalten, asynchrone Operationen / Aufgaben durchzuführen.

Worin besteht die AJAX-Asynchronie genau? Es ist die Fähigkeit von Javascript, eine Script- oder Element-API auf dem Server aufzurufen und nicht auf die Antwort zu warten. Stattdessen wird die Codeausführung fortgesetzt (z.B. kann der Benutzer die Seite sehen und mit ihr arbeiten). Wenn die Antwort kommt, wird die Ausführung des Code-Kontextes gestoppt (früher oder später, je nach Priorität der Aufgabe) und es gibt einen Rückruf (es handelt sich um eine Funktion, die ausgeführt wird, wenn der Server eine Antwort liefert). Ein solcher Code oder eine solche Funktion wird als "non-blocking" bezeichnet, da sie den Ablauf des Skripts nicht blockiert, indem sie auf die Antwort oder Verarbeitung wartet.

AJAX ist eine Methode zur Programmierung in Javascript. Es handelt sich nicht um eine neue Sprache oder ein neues Framework oder eine Bibliothek eines Drittanbieters. Es ist eine Methode, um die Daten in der Anwendung mit einer Datenbank, Server-Skripten (PHP, Java, ASP, etc.) auszutauschen, ohne die gesamte Seite aktualisieren / neu laden zu müssen. Technisch gesehen aktualisieren wir keinen Teil der Anwendung oder eine Seite (in Bezug auf die Aktualisierung) - das ist eigentlich die AJAX-Essenz.

Warum ist XML enthalten? Es ist eines der Datenformate, in denen AJAX die Informationen vom Server abrufen kann. Die bekanntesten und am häufigsten verwendeten sind JSON, XML, Text, Binärdaten. Jeder von ihnen kann auf eine andere Weise verwendet werden.

AJAX ermöglicht es, die vom Benutzer eingegebenen Daten zu überprüfen, noch bevor der Benutzer sie durch Bestätigen des Formulars sendet. Bis zu einem gewissen Grad kann dies auch per Javascript geschehen, aber bei einem Anmeldeformular mit bedingtem Passwort, Benutzername oder E-Mail, die innerhalb einer Tabelle oder Datenbank

eindeutig sein müssen, sind dies genau die Situationen, für die AJAX die richtige Technologie ist. Es ermöglicht die gleichen Operationen, die auch durch den Aufruf von PHP-Skripten beim Umleiten nach dem Senden des Formulars ausgeführt werden können, aber im Gegensatz zu ihnen können Sie diesen Aufruf im Hintergrund ausführen.

AJAX Grundlagen

Die Grundidee von AJAX besteht darin, dass Teile einer Webseite asynchron hochgeladen werden, was den Inhalt der Seite verändert. Das bedeutet, dass eine Änderung des Inhalts nicht zum Hochladen der gesamten Seite führt, sondern nur des geänderten Inhalts. Zu diesem Zweck wird das Objekt XMLHttpRequest aus JavaScript aufgerufen. Diese Anforderung fragt den Server nach dem Inhalt, der typischerweise mit XML und JavaScript zurückgesetzt wird. Das Skript analysiert dann dieses XML. Die Daten aus dem XML-Skript übersetzen und ändern dann die Webseite mit DOM (Document Object Model) entsprechend.

Die Grundprinzipien der Arbeit mit AJAX werden an einem praktischen Beispiel beschrieben - einem Diskussionsforum, in dem die Diskussionen in HTML als Baum dargestellt werden. Wenn die Nutzer einen Beitrag lesen wollen, müssen sie auf dessen Überschrift klicken. Bei einem klassischen Modell wird die Anfrage an den Server gesendet und anschließend eine ganze Seite erstellt, die den Text des Beitrags und den gesamten "Baum" enthält. Im Falle des AJAX-Modells wird als Reaktion auf die Anfrage nur der Text des Beitrags gesendet. Das Hinzufügen des Textes zur ursprünglichen Webseite wird durch ein entsprechendes JavaScript-Programm sichergestellt.

Für ältere Internet Explorer-Versionen (ab Version 5.0) stand für diese Aufgabe ein ActiveX-Objekt zur Verfügung. Wie andere Entdecker stellt auch IEZ das entsprechende Objekt nativ zur Verfügung. Die aktuelle API XMLHttpRequest des Internet Explorers ist unter MSDN verfügbar; Mozilla-Projekt-API ist ebenfalls verfügbar.

AJAX´s eigener Trick ist XMLHttpRequest - er kann asynchron verarbeitet werden, was in diesem Zusammenhang asynchron zur Anordnung des Restes der Seite sowie zu den laufenden Skripten bedeutet. Das bedeutet, dass die Daten über HTTP heruntergeladen werden können und dass die Daten heruntergeladen werden können, ohne die Interaktion des Benutzers mit der Seite zu beeinträchtigen. Die angeforderte Seite wird ebenfalls im Hintergrund erstellt.

Was ist AJAX? Einfach gesagt, ist AJAX eine Technologie, die Skripte verwendet, um die Kommunikation zwischen einer Seite und einem Webserver zu erleichtern. Auf diese Weise ist es möglich, den Seiteninhalt zu ändern, ohne ihn erneut hochladen zu müssen.

Theoretisch ist es möglich, dynamische Seiten ohne Serverskripte zu erstellen, d.h. eine Seite für das gesamte Web zu haben, die nur den Inhalt ändert. Dies ist jedoch nicht empfehlenswert - wie bereits gesagt wurde, handelt es sich um eine Skript-Technologie, so dass es vom Client abhängt und nicht gesichert werden kann, dass es funktioniert.

Dennoch kann es als Add-on für Webseiten verwendet werden - z.B. Meinungsumfrage oder ein Suchassistent wie bei Seznam.cz.

Wenn Sie sicher sind, dass AJAX für die Besucher Ihrer Webseiten funktioniert, kann es auch für komplexere Anwendungen verwendet werden. Es kann z.B. für einen Chat verwendet werden, der nur neue Beiträge herunterlädt; wenn der Besucher allein ist, verlangsamt sich die Suche nach Updates, was viele übertragene Daten speichert.

12. Frameworks

Framework (Application Framework) ist eine Softwarestruktur, die als Unterstützung bei der Programmierung und Entwicklung und Organisation anderer Softwareprojekte dient. Es kann Rahmenprogramme, API-Bibliotheken, Unterstützung von Entwurfsmustern oder empfohlene Verfahren in der Entwicklung enthalten.

- Zweck
- Architektur
- Beispiele
- Ähnliche Artikel
- Externe Links

Zweck

Ziel eines Frameworks ist es, typische Probleme eines bestimmten Bereichs zu übernehmen und so die Entwicklung zu erleichtern, damit sich die Designer und Entwickler auf ihre Aufgaben konzentrieren können. So kann sich beispielsweise das Team, das Apache Struts für die Entwicklung von Webseiten für eine Bank einsetzt, auf den Bankbetrieb konzentrieren und nicht auch auf eine perfekte Navigation zwischen den einzelnen Seiten.

Es wird argumentiert, dass die Verwendung von Frameworks den Code langsam oder ineffizient macht und dass die durch die Verwendung eines fremden Codes eingesparte Zeit für das Studium des Frameworks aufgewendet werden muss. Durch die Wiederverwendung oder den Einsatz für ein Großprojekt wird jedoch eine erhebliche Zeitersparnis erzielt. Nach der Deinstallation des Frameworks können einige Anwendungen nicht mehr funktionieren.

Architektur

Frameworks bestehen aus sogenannten frozen spots und hot spots. Frozen Spots definieren die Gesamtarchitektur der Softwarestruktur, ihre Grundkomponenten und die Beziehungen zwischen ihnen. Diese Teile ändern sich in keiner Weise in der Rahmennutzung. Im Gegenteil, Hot Spots sind Komponenten, die zusammen mit dem Code programmer´s eine ganz spezifische Funktionalität schaffen und daher jedes Mal anders sind.

In einer objektorientierten Umgebung besteht das Framework aus abstrakten und klasschen (nicht abstrakten) Klassen. Frozen Spots können durch abstrakte Klassen dargestellt werden und der Code (Hot Spots) selbst wird durch die Implementierung abstrakter Methoden hinzugefügt.

Framework ist eine Software-Struktur, die als Unterstützung bei der Programmierung

und Entwicklung und Organisation anderer Softwareprojekte dient. Es kann unterstützende Programme, API-Bibliotheken, Konstruktionsmuster oder empfohlene Verfahren in der Entwicklung beinhalten.

Ein weiterer sehr wichtiger Faktor bei der Wahl ist, wofür der Rahmen und sein Zweck benötigt werden. Im Allgemeinen können Frameworks in zwei Gruppen eingeteilt werden:

- Skriptsätze - Bibliotheken - für alle Arten von Bedürfnissen
- Skripte, die eine konkrete Webanwendung erstellen.

Vorteile

- Schnellere Entwicklung
- Weniger Code
- Universeller Code - Änderungen oder neue Funktionen werden viel einfacher hinzugefügt.
- Nette URL

Die Entwicklungsgeschwindigkeit ist höher. Framework erleichtert die Arbeit und vermeidet Programmierroutinen wie die Verbindung zu einer Datenbank, die Überprüfung der richtigen Option mit einem Abschnitt oder die Validierung eines Formulars. Cool-Url, XSS, separate Anwendungs- und Präsentationslogik, der Wechsel des DB-Servers von MySQL auf PostgreSQL wird ganz einfach sein.

Dank des Frameworks ist es möglich, sich nur auf die Entwicklung zu konzentrieren, aber es wird durch die Zeit eingelöst, die Sie benötigen, um zu lernen, mit Frameworks zu arbeiten. Wenn Sie lernen, etwas zu tun, werden Sie bald feststellen, dass es viel einfacher und schneller geht. Das Erlernen der Arbeit mit dem Framework kann eine Angelegenheit von einem Monat sein, aber auch eine Angelegenheit von mehreren Monaten oder einem Jahr. Schließlich werden Sie das Framework um Bibliotheken erweitern und die Programmierzeit deutlich verkürzen können. Framework bedeutet Zeit- und Kostenersparnis

Die Wahl eines Frameworks ist nicht einfach. Es wird empfohlen, mindestens zwei zu haben - eine für einfache und eine zweite für komplexere Aufgaben. Vor jedem Projekt muss entschieden werden, welches davon am besten geeignet ist. Natürlich ist das Beste, was man tun kann, diejenigen mit einer guten großen Gemeinschaft zu lernen, diejenigen, bei denen man sicher ist, dass ihre Entwicklung nicht in einer Woche abgeschlossen sein wird.

- **CakePHP:** ziemlich einfach zu erlernen; Qualitäts-Community; lange Entwicklung einer neuen Version
- **Zend Framework:** sehr effizient; deckt alle Anforderungen bei der Erstellung von

Webanwendungen ab; große Community; anspruchsvoll; lange Titel;
- **CodeIgniter:** klein, einfach; entwickelt von einer Firma, nicht von einer Community; besser als Cake in einigen Bereichen x "dumm" in anderen (es hat keine Layouts)

Monolithische Frameworks entkoppeln sich allmählich in einzelne Komponenten, was viele Vorteile mit sich bringt. War es früher schwierig und manchmal sogar unmöglich, nur eine Rahmenkomponente zu verwenden, ist es heute möglich, diese Komponente einfach zu installieren. Der Entwicklungszyklus der einzelnen Komponenten ist unterschiedlich. Sie können ihre eigenen Repositories haben, Issue Tracker, Entwicklungsteams.

Komponenten können keine neuen Versionen kontinuierlich aktualisiert werden, ohne auf eine andere Version des gesamten Frameworks zu warten. Es ist auch möglich, eine Komponente nicht zu aktualisieren, z.B. wegen eines BC-Bruchs.

Die Bedeutung des Begriffs "Framework" verschiebt sich also, und "Versionen" haben nun fast nicht mehr ihre ursprüngliche Bedeutung. Anstelle eines Frameworks XYZ, Version 2.3.1., wird ein Satz von Komponenten in verschiedenen Versionen verwendet, die zusammenwirken.

12.1. Was ist ein Framework?

Bei der Erstellung moderner Webanwendungen werden häufig Frameworks oder Entwicklungsrahmen verwendet. Es handelt sich um eine Reihe von Bibliotheken und Quellcode, die wiederholt verwendet werden können, um die Arbeit zu erleichtern, und deren Funktionalität einen Teil der erstellten Anwendung abdecken kann. Frameworks können Probleme lösen, die in irgendeiner Weise verallgemeinert werden können; sonst wäre es nicht sinnvoll, einen Rahmen zu schaffen. Dies ermöglicht es dem Anwendungsentwickler, sich auf die Funktionen zu konzentrieren, die für die Anwendung einzigartig und exklusiv sind und keine Routineprobleme lösen müssen.

Verwendung von Javascript-Frameworks

Frameworks können für die meisten Programmiersprachen erstellt werden. Javascript Framework ist somit ein Framework, das verwendet wird, um die Arbeit und Programmierung in Javascript zu erleichtern. Die Verwendung von Frameworks ist sehr umfangreich und jeden Tag ist das Portfolio der Probleme, die Javascript-Frameworks lösen können, größer. Generell kann man sagen, dass sie für das effektive Schreiben von Quellcode verwendet werden können; sie lösen weitgehend auch die Inkompatibilität zwischen Webbrowsern und einem geschriebenen Code, wodurch die Zeit auf programmer´s eingespart wird und es möglich wird, Elemente zu verwenden, die schwer zu programmieren wären. Ihre Stärke und Nutzung ist vor allem mit der Verwendung der

AJAX-Technologie verbunden, da sie die Interaktion von Benutzer und Anwendung durch asynchrone Kommunikation zwischen Client und Server verbessern. Darüber hinaus ermöglichen sie den dynamischen und einfachen Zugriff und die Änderung der einzelnen Elemente einer Seite (DOM-Modell), GUI-Elemente, weisen Ereignisse zu und animieren sie. Last but not least enthalten einige Frameworks vorgefertigte GUI-Komponenten, die entweder nicht mit einer anderen Technologie implementiert werden können oder zu schwierig und ineffizient sind. Diese Elemente sind einfach zu bearbeiten und in einem Framework mit nur mehreren Codezeilen zu implementieren.

13. Die 10 besten PHP-Frameworks für Entwickler

PHP, weltweit bekannt als die beliebteste serverseitige Skriptsprache, hat sich stark weiterentwickelt, da in statischen HTML-Dateien erste Inline-Codefragmente auftauchten.

Damals mussten Entwickler komplexe Webs und Webanwendungen erstellen; bei einer gewissen Komplexität war es zu zeitaufwändig und es erforderte zu viel Aufwand, immer bei Null anzufangen. Daher erschien die Notwendigkeit einer strukturierteren und natürlicheren Art der Entwicklung. PHP-Frameworks bieten Entwicklern eine adäquate Lösung für dieses Problem.

Warum man das PHP-Framework verwenden sollte

Zunächst werden wir uns mit den stärksten Gründen befassen, warum viele Entwickler PHP-Frameworks bevorzugen und wie diese den Entwicklungsprozess verbessern. Die Vorteile von PHP-Frameworks sind wie folgt:

- Ermöglicht eine schnelle Entwicklung
- Bereitstellung von gut organisiertem, wiederverwendbarem und wartbarem Code
- Ermöglicht Wachstum im Zeitablauf, da Framework-basierte Webanwendungen skalierbar sind.
- Keine Sorgen um Low-Level-Websicherheit
- Nach MVC (Model-View-Controller) Muster, das die Trennung von Darstellung und Logik gewährleistet.
- Förderung moderner Webentwicklungspraktiken einschließlich objektorientierter Programmierwerkzeuge

1. Larve

   Obwohl Laravel ein relativ neues PHP-Framework ist (es wurde 2011 veröffentlicht), ist es laut der neuesten Online-Umfrage auf Sitepoint das beliebteste Framework bei Entwicklern. Laravel verfügt über ein riesiges Ökosystem mit einer Plattform, die sofort gehostet und bereitgestellt werden kann, und das offizielle Web bietet viele Tutorials in Form von Screencasts namens Laracasts.

2. Symfony

   Die Komponenten des Frameworks Symfony 2 werden von vielen Projekten genutzt, wie z.B. dem System zur Verwaltung der Inhalte von Drupal oder der Software phpBB für laufende Foren. Es wird auch von Laravel verwendet. Symfony hat eine umfang-

reiche developers´ Community und viele Unterstützer.

3. CodeIgniter

CodeIgniter ist ein fast 10-jähriges, leichtes PHP-Framework (ursprünglich 2006 veröffentlicht). CodeIgniter hat einen sehr einfachen Installationsprozess, der nur eine minimale Konfiguration erfordert. Es ist eine ideale Option, um Konflikte mit PHP-Versionen zu vermeiden, da es auf fast allen Shared und Dedicated Hosting Plattformen reibungslos funktioniert (derzeit benötigt es nur PHP 5.2.4).

4. Yii 2

Die Wahl von Yii-Frameworks gibt einer Seite einen Leistungsschub, da sie schneller ist als andere PHP-Frameworks. Es nutzt weitgehend die Technologie des "faulen Ladens". Yii 2 ist rein objektorientiert und basiert auf dem DRY (Don't Repeat Yourself) Codierungskonzept, so dass es eine klare und logische Codebasis bietet.

5. Phalcon

Das Phalcon-Framework wurde 2012 veröffentlicht und wurde bald bei den PHP-Entwicklern beliebt. Es soll so schnell wie ein Falke sein, wie es in C- und C++-Sprachen geschrieben wurde, um eine höchstmögliche Leistungsoptimierung zu erreichen. Für die Verwendung von Phalcon ist es nicht notwendig, die Sprache C zu lernen, da die Funktionalität als PHP-Klassen bereitgestellt wird, die in jeder Anwendung verwendet werden können.

6. CakePHP

CakePHP wird seit einem Jahrzehnt verwendet (die erste Version wurde 2005 veröffentlicht), aber es ist immer noch eines der beliebtesten PHP-Frameworks, da es immer bestrebt war, aktuell zu sein. Die neueste Version, CakePHP 3.0, bietet eine erweiterte Sitzungsverwaltung, verbesserte Modularität durch die Trennung mehrerer Komponenten und erhöhte die Möglichkeit, mehr autarke Bibliotheken zu erstellen.

7. Zend Framework

Zend ist ein robustes und stabiles PHP-Framework mit einer Reihe von Konfigurationsoptionen; daher wird es normalerweise nicht für kleinere Projekte empfohlen. Es ist jedoch großartig für die komplexeren. Zu den Zend-Partnern gehören z.B. IBM, Microsoft, Google und Adobe. Die kommende neue Version, Zend Framework 3, wird für PHP 7 optimiert, aber sie wird weiterhin PHP 5.5 unterstützen.

8. Schlank

Slim ist ein PHP-Mikroframework, das nur das bereitstellt, was benötigt wird. Das Design von Micro frameworks´ ist minimalistisch, sie sind ideal für kleinere Anwendungen, für die ein voll ausgestattetes Framework zu viel wäre. Der Slim Schöpfer wurde von einem Mikrorahmen Rubin namens Sinatra inspiriert.

9. FuelPHP

FuelPHP ist ein flexibles, voll funktionsfähiges PHP-Framework, das nicht nur MVC, sondern auch die entwickelte Version HMVC (Hierarchical Model-View-Controller) unterstützt. FuelPHP fügt eine optionale Klasse namens Presenter (früher ViewModel genannt) zwischen den Ebenen View und Controller hinzu, um die für die Erzeugung von Views erforderliche Logik einzubinden.

10. PHPixie

PHPixie ist ein brandneues Framework, das 2012 gestartet wurde, um ein leistungsfähiges Framework für schreibgeschützte Webs zu schaffen. Wie FuelPHP implementiert PHPixie das Entwurfsmuster HMC und wird mit Hilfe unabhängiger Komponenten erstellt, die auch ohne das Framework verwendet werden können. PHPixies-Komponenten werden zu 100% durch Unit-Tests getestet und erfordern nur ein Minimum an Abhängigkeiten.

WEB-TECHNOLOGIEN

1. HTML

HTML oder **HyperText Mark-up Language** ist ein Werkzeug zur Erstellung von Webseiten - die aktuelle Version ist - HTML5 (HTML 5.2). Diese Sprache ist durch die Organisation W3C standardisiert.

Besteht aus Tags und Attributen.

Die HTML-Anzeige erfolgt über einen Browser und in mehreren Schritten (= Parsen).

- Der Browser ruft das Dokument ab und führt eine DTD-Syntaxanalyse durch.
- Jedem Element ist ein Stil zugeordnet (Anzeigemodus).
- Anwendung von Code in Skriptsprachen
- Schritt-für-Schritt-Rendering der Seite

HTML-Tags werden nach Bedeutung unterteilt:

- Strukturale, welche das Dokument strukturieren. Beispiel: Absatz (< p >), Überschriften (< h1 >, < h2 > .....) ....
- Beschreibend (semantisch), beschreibt die Art des Inhalts des Elements, z.B: Titel (< title >), Adresse (< address >) .....
- Stilistik, die z.B. das Aussehen eines Elements bei der Betrachtung bestimmt: Fett (< b >), kursiv (< i >), Hervorhebung (< strong >) .....

Jedes HTML-Dokument sollte dem **Grundschema** folgen:

Am Anfang steht die DTD-Direktive, die Document Type Declaration:

```
<! DOCTYPE html >
```

Folgt dem Element:

```
< html >    </ html >
```

Die Überschrift eines Dokuments (enthält Metadaten):

```
<head>
      <meta charset = ”utf-8”> - Kodierung
      <title > Seitenüberschrift </ title >
```

Autor, Beschreibung, Schlüsselwörter, kaskadierende Stile.....

```
< head >
```

Hinter der Kopfzeile folgt der Textkörper des Dokuments, der den Inhalt der Seite als Text, Bilder, Links, Tabellen etc. zusammenfasst.

```
<body> … </body>
```

Um HTML zu schreiben, ist es ratsam, einen Texteditor zu verwenden, der die Farbsyntax (Farbcodierung für einzelne Teile des Codes - Tags, Eigenschaften, Klartext) leitet, d.h. den Marker anspricht, die Tabs kennt oder es schafft, das Dokument gemäß der vorgegebenen Spezifikation zu validieren.

Zum Beispiel: Notepad++, PSPad...

Alternativ können Sie auch WYSIWYG (What You See Is What You Get) Editoren verwenden, die direkt auf die fertige Seite verweisen. Der Benutzer muss HTML nicht kennen - nur die Seite einfügen und der Seiteneditor generiert den entsprechenden Code.
Zum Beispiel: Adobe Dreamweaver, Microsoft Expression Web, LibreOffice Writer / Web, Bluegrifon

2. CSS - Kaskadierende Stile

CSS ist eine Sprache zur Beschreibung der Darstellung von Elementen auf Seiten, die in HTML, XHTML oder XML geschrieben sind. Sie sind entwickelt und standardisiert vom W3C. Die aktuelle Version ist CSS 3.

CSS kann auf verschiedene Weise mit dem HTML-Code verbunden werden:

- Im HTML-Seitencode mit dem Element u < style > </ style>.

```
<style = ”text/css ”>
      # head {
            width : 200px;
            height : 450px;
      }
</style>
```

- Hilfe bei der externen Datei - Element <link>

```
<head>
      <link rel = ‘style sheet’ href = ‘styles.css’ type =
      ’text/css ’>
<head>
```

- Direkte Inline-Einschreibung über das Attribut style

```
<p style = ” color : red ; text- decoration : underline ”>
Dieser Absatz wird rot und unterstrichen dargestellt. </ p>
```

Die PHP-Sprache besteht aus Regeln, wobei jede Regel einen Selektor und einen Deklarationsblock enthält. Jeder Deklarationsblock enthält dann einzelne Deklarationen, die durch ein Semikolon getrennt sind. Jede Deklaration besteht aus Eigenschaftsbezeichner, Doppelpunkten und Werten. Es kann noch mit dem Tag !important versehen werden, was die Kraft der Deklaration erhöht.

Warum CSS statt HTML-Formatierung verwenden?

CSS bietet:

- weitere Formatierungsoptionen, die HTML nicht bietet - z.B.: Bestimmung des Abstands der Elemente vom Seitenrand aus.
- einfacheres Bearbeiten Ihres Erscheinungsbildes - Ändern Sie die Farbe aller Überschriften auf einmal, ändern Sie die Schriftart,....
- die Möglichkeit, eine Vorlage für mehrere Seiten auf einmal zu erstellen.
- die Trennung von Struktur und Stil - im HTML-Inhalt, im CSS-Auftritt
- Caching-Stile - Schnellerer Seitenabruf (aber zusätzlich eine HTTP-Anfrage zum Laden einer externen Datei)
- Dynamische Änderungen an CSS-Eigenschaften mit JavaScript
- Mit CSS kann jede XML-Sprache formatiert werden.
- Anzeigeeinstellungen für einzelne Geräte - bedingtes CSS

Der Endbenutzer kann seinen eigenen CSS-Stil für jede Seite schreiben - Sie können alle Links auf jeder Seite so einstellen, dass sie immer unterstrichen werden, oder dass die Schriftart auf dieser speziellen Seite immer schwarz ist.

In Kombination mit JavaScript können effektive Lesezeichen erstellt werden, die das Erscheinungsbild der Seite verbessern können. Entfernen Sie beispielsweise alle Hintergrundbilder, ändern Sie den Hintergrund auf Weiß und die Schriftart auf Schwarz, etc.

CSS hat auch einige Nachteile. Teilweise wird es von den meist genutzten Browsern nicht unterstützt oder es treten Fehler bei der Implementierung von CSS in Browsern auf. Dies kann durch die Verwendung verschiedener Stile für verschiedene Browser umgangen werden.

Bedingte Kommentare können verwendet werden, um verschiedene Browser einzustellen.

```
<!--[if IE]> <style type="text/css"> # alert {color: blue;}
</style> <![endif]-->
```

Dieser Code wird nur vom Internet Explorer interpretiert, andere Browser sehen einen gemeinsamen HTML-Kommentar und interpretieren das interne Stylesheet nicht.

Auch CSS hat auch einige **Grenzen**:

- CSS-Selektoren bieten keinen Zugriff auf übergeordnete Elemente.
    - Beispielsweise können Sie nicht einfach die Absätze streichen, die einen Link enthalten.

- Die horizontale Steuerung der Elemente auf der Seite ist intuitiv und einfach, während das vertikale Styling einen umfassenderen Ansatz erfordert (z.B.: Flexbox oder Raster).

- CSS bietet keine Möglichkeit, Variablen oder Konstanten symbolisch zu schreiben.

- Alle Werte müssen direkt in den Code geschrieben werden.
    - Wird beispielsweise die gleiche Farbe an mehreren Stellen verwendet, kann man nicht die symbolische Beschriftung farbe = rot verwenden und anschließend einfach die Farbvariable schreiben. Stattdessen muss der Rotwert überall eingegeben werden. Diese Einschränkung entfernen CSS-Preprocessor (wie SASS, LESS, Stylus).

- CSS2 bietet keine Möglichkeit, runde Rahmen oder andere runde Objekte zu erstellen, sondern funktioniert nur mit Rechtecken.

- CSS2 bietet keine Möglichkeit einem Element mehrere Hintergrundbilder zuzuordnen.

3. Kaskadierende Styles - Fortgeschrittene Techniken

Es ist wichtig zu wissen, welche version von CSS welche Funktionen bietet und definiert.

CSS Version 1 führt ein die in die Syntax, welche ebenfalls in den folgenden Versionen verwendet wird und bieten die Möglichkeit, Elemente über Selektoren auszuwählen, Pseudoklassen und definiert Werte und deren Einheiten.

Kategorien, für die CSS 1 Werte und Einheiten definiert:

- Schriftarteneigenschaften
- Textfarben und Hintergründe
- Texteigenschaften
- Eigenschaften von Blockelementen
- Wie man Elemente anzeigt
- Positionsmanagement

Werte und Einheiten

Für Dezimalwerte wird ein Dezimalpunkt verwendet.

Längeneinheiten

- (keine) - für dimensionslose Eigenschaften (z.B. Linienhöhe)
- % - Prozent, Einheit bezogen auf die implizite Dimension, geschrieben ohne Leerzeichen
- pt - typographischer Punkt, Standardeinheit ist 1/72 Zoll.
- px - 0.75 pt
- PC - pica, 1 PC = 12 pt
- cm - Zentimeter
- mm - Millimeter
- in - Zoll
- em - square, ist gleich der Basis-Schrifthöhe.
- ex - die Höhe des Buchstabens "x" ist relativ zur verwendeten Schriftart.

Farben in CSS werden über die RGB-Palette eingegeben und so weiter:

- Entweder numerisch # rrggbb - zweistelliger Hexadezimalwert (00 bis ff) - 16 Millionen Farben
- # rgb - Hexadezimale einstellige Zahl (0 bis f) - 4.096 Farben.
- Als sichere Farbe, die eine noch kleinere Teilmenge von Farben ist: Dies sind ein-

stellige Farben, deren Werte um 3 steigen - Kombinationen von sechs Werten {0, 3, 6, 9, C, F} = 216 Farben.
- rgb (r, g, b) - in Dezimalzahlen (0 bis 255) und Syntax als Funktion - 16 Millionen Farben
- rgb (r%, g%, b %) - Werte im Bereich (0 bis 100)

Alternativ können Sie auch vordefinierte Konstanten verwenden, die auf textbasierten Namen mit dem Namen Farben basieren. Z.B. Aqua (helles Blaugrün), Schwarz (Schwarz) = # 000, Blau, Fuchsia (Anilinrot), Grau, Grün, Limette, Kastanienbraun, Marine, Olive, Lila, Rot, Silber, Blaugrün, Weiß (Weiß) = #FFF, Gelb. In der CSS-Version 4.0 sind 148 Farben benannt.

Die Schriftart in CSS 1 hat ihre Attribute.

- Schriftart: normal, kursiv (kursiv), schräg (geneigt römisch)
- Schriftgröße: mittel; kleinere Größen: xx-klein, x-klein, klein, klein, klein; größere Größe: groß, x-groß, xx-large, kleiner, größer, Größe in Prozent 100% ist die Standardgröße.
- Schriftart: Fettheit: normal, fett, fetter, dicker, Sie können die Zahl: 100, 200 eingeben..... 900 (400 = normal, 700 = fett)
- Schriftvariante: Kapitälchen, normal
- Schriftdekoration: unterstrichen (unterstrichen), überstrichen (überstrichen), blinkend (blinkend), line-through (durchgestrichen) Keine (Standard)

CSS 1 führt auch gängige Schriftarten ein.

- Serifen - klassische Schriftart (z.B. Times New Roman)
- sans - serif - sans serif (z.B. Helvetica oder Arial)
- kursiv - italic
- fantasy - dekorative Schriftart
- monospace - Nicht-proportionale Schrift (z.B. Courier)

Als nächstes definiert CSS 1 das URL-Element so, dass:

- construction url (), wobei die Quelladresse in Klammern angegeben wird
- Absolute: url (http://www.example.com/ images / logo.png)
- Relativ zum Server: URL (/ Bilder / logo.jpg)
- Bezogen auf das aktuelle Verzeichnis: url ("images /logo.jpg")

Wenn die URL Kommas, Leerzeichen, Anführungszeichen oder das Ende einer Klammer enthält, können diese Zeichen durch einen Backslash escaped werden.

Dann Selektoren:

- Typenauswahl: A - Alle Elemente des Typs A
- Klassenauswahl: A. Klasse - Alle Elemente A mit Attributklasse = "Klasse".
- ID-Auswahl: A # ID - Alle Elemente mit ID
- Follower Auswahl: AB - Alle B-Elemente innerhalb von A

Die **Definition von CCS 2** hat weitere Definitionen gebracht:

- Umrisse - Außengrenzen
- maximale Höhe, maximale Breite, minimale Höhe, minimale Breite - minimale und maximale Breite oder Höhe des Elements
- Inhalt - einstellbarer Elementgehalt
- Zähler - Kapitel-Nummerierungswerkzeug
- Zitate - Zitat-Stil
- clip - zuschneiden
- cursor - cursor über das Element
- Position - Möglichkeit, ein Element in einer Reihe, in einem Block, absolut, relativ,.... zu positionieren.
- oben, unten, rechts, links Randwerte für Position: absolut ;
- overflow - Überlaufansicht
- Sichtbarkeit - Sichtbarkeit der Elemente
- z-index - Überlappungsmöglichkeiten
- Seitenumbruch, Waisen, Witwen - typografische Regeln für Seitenumbrüche
- Tabellenlayout, Randkollaps, Randabstand, Beschriftungsseite, Leerzellen - Neue Optionen zur Anzeige von Tabellen
- Richtung – Schreibrichtung

CSS 2 ermöglicht es Ihnen, neue Arten von Selektoren zu verwenden:

- Universell einsetzbar: * Gilt für alle Elemente
- Kind-Auswahl: A> B - berücksichtigt nur die Elemente B, die direkt mit A verschachtelt sind.
- Ein Geschwisterwähler: A + B - wählt alle B-Elemente aus, die den gleichen Elternteil wie A haben und direkt darauf folgen.

Es definiert auch Selektoren über Attribute:

- A [attr] - alle Elemente A, die eine attr-Attributmenge haben.
- A [attr = value] - alle Elemente A, die das Attribut attr = "value" haben.
- A [attr ~ = value] - alle Elemente A mit einem Attribut “attr”, dessen Wert eine Liste von Wörtern ist, die durch ein Leerzeichen getrennt sind, und nur eines dieser Wörter ist identisch mit "value".
- A [attr | = value] - alle Elemente A, deren Wert des Attributs “attr” mit der Zeichenkette "value" beginnt, dann der Bindestrich und die nächste Zeichenkette.

Es führt auch ein Pseudoelement- und Pseudoklassensystem ein.

- Pseudoelemente
  - Erste Zeile - :first-line
  - Erster Buchstabe (Initialen) - :first-letter
  - Vorher - :before
  - Für - :after

- Sprach-Pseudoklasse
  - : lang

- Elterliche Pseudoklasse
  - Erster Nachkomme - :first-child

- Pseudo-Klassen-Links
  - Unbesuchter Link - :link
  - Besuchter Link - :visited
  - Fokussierter Link - :focus
  - Maus-Link - :hover
  - Aktiver Link - :active

Damit alles richtig funktioniert, muss die Reihenfolge der Definitionen eingehalten werden. Jede Pseudo-Klasse hat eine andere Priorität.

Schließlich bietet die neueste Version von CSS3, die mit HTML5 verbunden ist:

- Animation - CSS3 unterstützt direkt die Animation von Elementen (deren Eigenschaften), Animationen wurden mit DHTML erstellt, z.B. jQuery
- zusätzliche Optionen für das Styling von Hintergründen für Blockelemente, einschließlich Zuschneiden von Hintergründen, Schlagschatten oder abgerundeten Kanten.
- zusätzliche Regeln für das Überlaufen des Inhalts von Blockelementen
- Opazität - der Grad der Opazität der Elemente
- zusätzliche Unterstützung für seitenbezogene Inhalte - Lesezeichen und Optionen zur Textaufteilung
- flexible Blockelemente
- Ziellinks - wie und wo man sie öffnet
- Eigenschaften für Drag'n'Drop
- Zusätzliche Attribute für Schriften - Abruf von Schriften aus einer externen Quelle, Größenanpassung bei schlechter Lesbarkeit, Schriftverengungen / Skalierung
- Eigenschaften für den generierten Inhalt - Inhaltskürzung bei Erweiterbarkeit, Verschieben von Elementen weiter auf der Seite
- Netze (noch nicht implementiert)
- neue Funktionen für marquee

- automatisches mehrspaltiges Layout
- Die neuen Funktionen von Ruby
- Eigenschaften für gelesene Texte
- 2D- und 3D-Transformationen
- Funktionen, die mit der Navigation funktionieren
- benutzerdefinierte Eigenschaften

4. JavaScript

JavaScript ist eine Programmiersprache, die auf Webseiten verwendet wird, direkt in HTML-Code geschrieben wird und zu den Client-Skripten gehört (auf der Client-Seite ausgeführt).

JavaScript ist:

- interpretiert - muss nicht kompiliert werden
- verwendet Browser-Objekte und eingebaute Objekte.
- Browser-abhängig - funktioniert in den meisten Browsern
- Groß-/Kleinschreibung beachten - abhängig von der Schriftgröße im Eintrag
- Syntax ähnlich wie bei C, Java usw.

JavaScript hat gewisse Einschränkungen:

- Funktioniert nur im Browser.
- Der Benutzer kann JavaScript deaktivieren.
- Es gibt verschiedene Versionen von Sprache und Browsern, was zu häufigen Fehlern führt.
- Der Zugriff auf Dateien (außer Cookies) oder Systemobjekte ist nicht möglich.
- Es können keine Daten (außer Cookies) gespeichert werden.
- Dennoch ist JavaScript weit verbreitet und kann oft als integrierte Skriptsprache für viele Anwendungen verwendet werden.

Es ist zu finden in:

- Die meisten Erweiterungen für Webbrowser
- Einige NoSQL-Datensätze wie MongoDB oder CouchDB akzeptieren Abfragen in JavaScript.
- Adobe - Acrobat und Adobe Reader, Werkzeuge in der Adobe Creative Suite (Photoshop, Illustrator, Dreamweaver und InDesign)
- Office-Suite von Anwendungen OpenOffice ermöglicht die Verwendung von JavaScript als Skriptsprache.
- Interaktive Verarbeitung des Max / MSP Musiksignals
- Die digitale Software der Apple Logic Pro X Audio Workstation ermöglicht es Ihnen, benutzerdefinierte MIDI-Effekt-Plug-Ins mit JavaScript zu erstellen.
- ECMAScript (JavaScript) wurde in den VRML97 VRML Scripting Standard aufgenommen.
- Die Spiele-Engine Unity 3D unterstützt eine modifizierte Version von JavaScript für das Skripting mit Mono.
- DX Studio (3D-Engine) verwendet die JavaScript-Implementierung SpiderMonkey für Spiele und die Simulation von Logik.
- Maxwell Render bietet die ECMA-Skripting-Engine für die Aufgabenautomatisie-

rung.

- Google Apps Script auf Google Tables und Google Sites ermöglicht es Benutzern, benutzerdefinierte Formeln zu erstellen, wiederkehrende Aufgaben zu automatisieren und auch mit anderen Produkten von Google als Gmail zu kommunizieren.
- SpinetiX-Produkte verwenden SpiderMonkey JavaScript für das Skripting in SVG-Dateien.

Javascript kann auch als Scripting-Engine verwendet werden:

- Aktives Technologie-Skripting von Microsoft
- Programmiersprache Java in der 6. Version präsentiert das Paket javax.script
- Das Tool Qt C ++ beinhaltet ein Modul QtScript, das JavaScript interpretiert, sowie ein Java-Paket javax.script.

JavaScript kann auf verschiedene Weise ähnlich wie CSS in HTML geschrieben werden.

Tag < script > kann verwendet werden, um das Skript direkt in den HTML-Stream zu schreiben.

```
< script >
        alert (”Kopf hoch, es wird noch schlimmer!”);
</ Skript >
```

Oder fügen Sie eine externe Skriptdatei an.

```
< script src = ”externes_skript.js”> </ script >
```

Eine weitere Möglichkeit ist die In-Line-Anmeldung.

```
<p> <A href =”#” onClick =” alert(‘Hello’);”>
Click Me </a> </ p>
```

Am häufigsten wird jedoch die kombinierte Nutzung genutzt. Externes Skript definiert Funktionen, normales Schreiben (Verwendung des <script>) Variablen werden initialisiert und Startfunktionen und Inline-Skripte rufen Funktionen entsprechend den Ereignissen in Abhängigkeit von den Reaktionen des Benutzers auf.

JavaScript wird auf Websites hauptsächlich zum Booten einer Seite verwendet - Unter-

scheidung von Browsern, Aufruf von Menüs aus einer Datei, Deklaration von Funktionen, document.write (), oder als Benutzer ein Ereignis wie das Übergeben eines Mauselements, Klicken, Ändern der Fenstergröße, Ausfüllen eines Formulars.

5. JavaScript - Fortsetzung

JavaScript ist eine objektorientierte Programmiersprache. Es unterstützt daher das klassische Objektmodell.

- object.method () - Aufruf der Methode (Funktion), Befehl, der etwas bewirkt.
- object.property - bezieht sich auf die Eigenschaft eines bestimmten Objekts, es hat einen Wert, tut aber nichts.
- object.subobject - Referenz auf ein verschachteltes Objekt

JavaScript hat Zugriff

- Zu Browser Objekten (aus der Klasse Windows)
- Zu den Elementen der Seite
- Zu Mathematik- und Datumsobjekten, strings
- Zu erstellten Objekte

Class Window (Windows-Objekt) ist der Höhepunkt der Hierarchie der Objekte (Klassen). Seine Unterteilungen sind:

- Ort - die Adresse des geladenen Dokuments
- Historie - Browserverlauf
- Navigator - Informationen über die Version und den Browsertyp
- Screen - Bildschirmeigenschaften (Breite, Höhe, Farbe)
- Frames - arbeiten mit Frames ("Frame", Frameset)
- Ereignis - Mausereignisse, Tastatur
- Dokument - Bilder, Formulare, Links, Farben, einzelne HTML-Elemente.

Die am häufigsten verwendete Klasse ist die Dokumentenklasse, mit der Sie HTML-Dokumente (und andere Arten von Dokumenten) manipulieren können.

Seine wichtigsten Methoden sind:

- document.images
- document.forms
- document.applets
- document.links
- document.anchors
- document.all
- document.frames
- document.styleSheets
- document.scripts
- document.selection

- document.getElementById()
- document.getElementsByTagName()

Die Klasse Math wird verwendet, um höhere Mathematik zu verwenden.

Enthält die folgenden Methoden:

- abs (x)
- exp (x)
- log (x)
- max (x, y)
- min (x, y)
- pow (a, x)
- random ()
- sqrt (x)
- ceil (x)
- floor (x)
- round (x)
- acos (x)
- asin (x)
- atan (x)
- atan2 (x, y)
- cos (x)
- sin (x)
- tan (x)
- Math.E
- Math.LN10
- Math.LN2
- Math.LOG10E
- Math.LOG2E
- Math.PI
- Math.SQRT2
- Math.SQRT1_2 - square root of 1/2

Die Datums-Klasse wird verwendet, um mit Datum und Uhrzeit zu arbeiten (verschiedene Countdowns erstellen, Kalender….).

Hauptmethoden der Datums-Klasse :

- get / setFullYear ()
- get / setMonth ()
- get / setDate ()
- get / setDay ()
- get / setHours ()

- get / setMinutes ()
- get / setSeconds ()
- get / setMilliseconds ()
- get / setTime ()

Die Klasse String arbeitet mit Textzeichenketten. Sie können eine Längeneigenschaft auf Textstellen anwenden, die die Zeichenkettenlänge zurückgibt.

Die wichtigsten Methoden für die Arbeit mit strings sind:

- toUpperCase ()
- toLowerCase ()
- toString ()
- charAt (n)
- charCodeAt (n)
- substring (a, b)
- substr (a, b)
- concat(string1, string2, stringN)
- fromCharCode(code1,code2,..., codeN)
- indexOf (substring)
- lastIndexOf (substring)
- split (separator)

6. XML und Json

XML oder **eXtensible Mark up Language** ist eine Markup-Sprache mit standardisiertem W3C. XML kann als der Standard des Formats für den Informationsaustausch mit internationaler Unterstützung und hohem Informationsgehalt bezeichnet werden. Es kann leicht in andere Formate konvertiert werden, es gibt eine automatische Dokumentstrukturprüfung und unterstützt Hypertext und Links.

Die XML-Effizienz ist stark abhängig von der Struktur. Mit einer schlecht gestalteten Struktur ist das XML-Dokument unlesbar und ineffizient.

Jedes XML-Dokument

- Muss genau ein Wurzelelement haben.
- Nicht leere Elemente müssen durch die Start- und Stoppmarken begrenzt werden.
- Leere Elemente können als "leeres Element" markiert werden.
- Alle Attributwerte müssen in Anführungszeichen eingeschlossen werden - einfach (') oder doppelt (“).
- Das entgegengesetzte Paar von Anführungszeichen kann innerhalb der Werte verwendet werden.
- Elemente können verschachtelt werden, können sich aber nicht überlappen, d.h. jedes (nicht wurzelnde) Element muss vollständig in einem anderen Element enthalten sein.

XML-Beispiel

```
<? xml version = ”1.0” encoding = ”UTF-8”?>      XML declaration
< directory >                                    Root element
< person >                                       Nested element 1
< name > Adam < / name >                         Nested Element 1.1
< phone > 777 777 777 < / phone >                Nested element 1.2
< email > adam@adam.com < / email >              Nested element 1.3
< / person >                              Exiting the nested element 1
< person >                                Nested element 2
< name > Bara < / name >                  Nested element 2.1
< phone > 666 999 666 < / phone >         Nested element 2.2
< email > bara@bara.com < / email >       Nested Element 2.3
< / person >                                    Exiting nested element 2
< / directory >                                 Exiting the root element
```

Json - JavaScript Object Notation - ist ein leichtes Format für den Datenaustausch, ein-

fach lesbar und beschreibbar für eine Person, im Textformat geschrieben und völlig unabhängig von der Sprache.

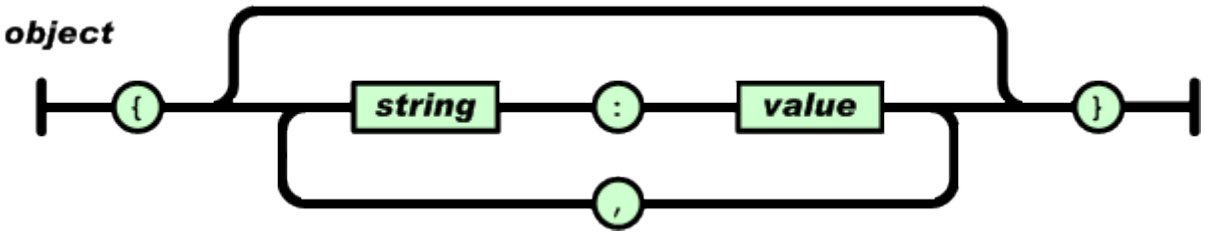
Es besteht aus zwei Strukturen

- Paarkollektion - Name / Wert
  - Realisierung: Objekt, Datensatz, Struktur, Dictionary, Hash-Tabelle, Keyed List, Assoziative Anordnung

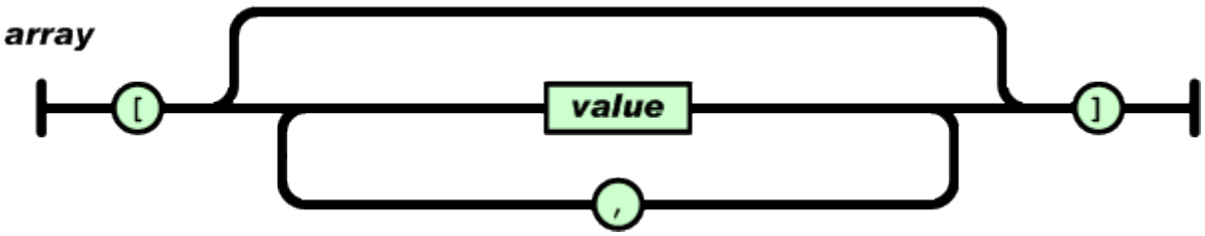
- Zugewiesene Werteliste
  - Realisierung: Array, Vektor, Liste, Sequenz, Sequenz

Objekt - Ungeordneter Satz von Paaren Name / Wert

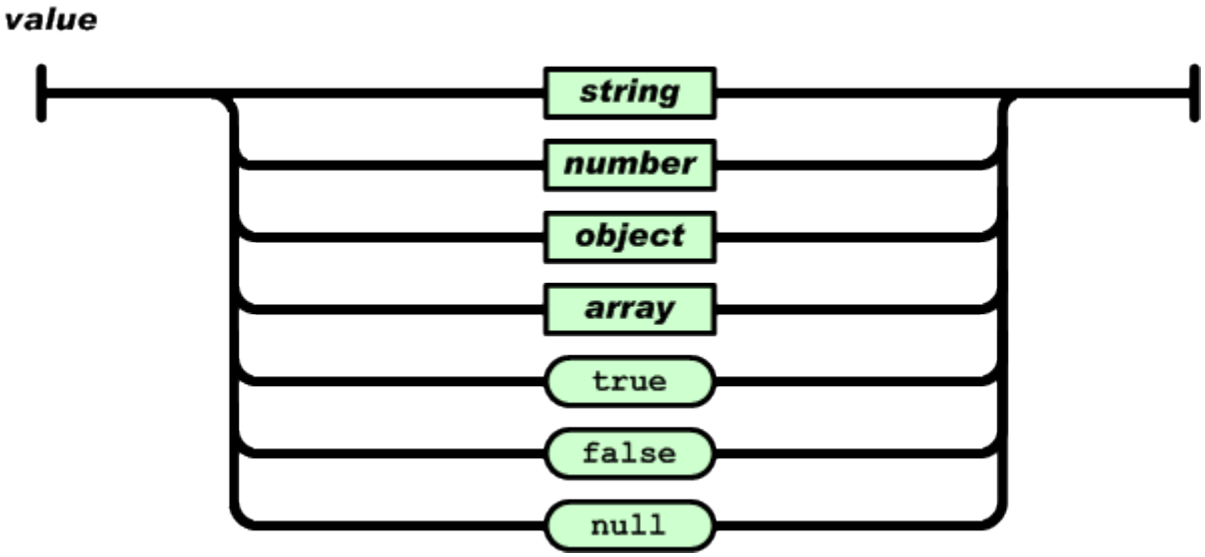
Geschrieben: {name1: value1, name2: value2}



Anordnung - Geordnete Wertsammlung
Geschrieben: [value1, value2, value3]

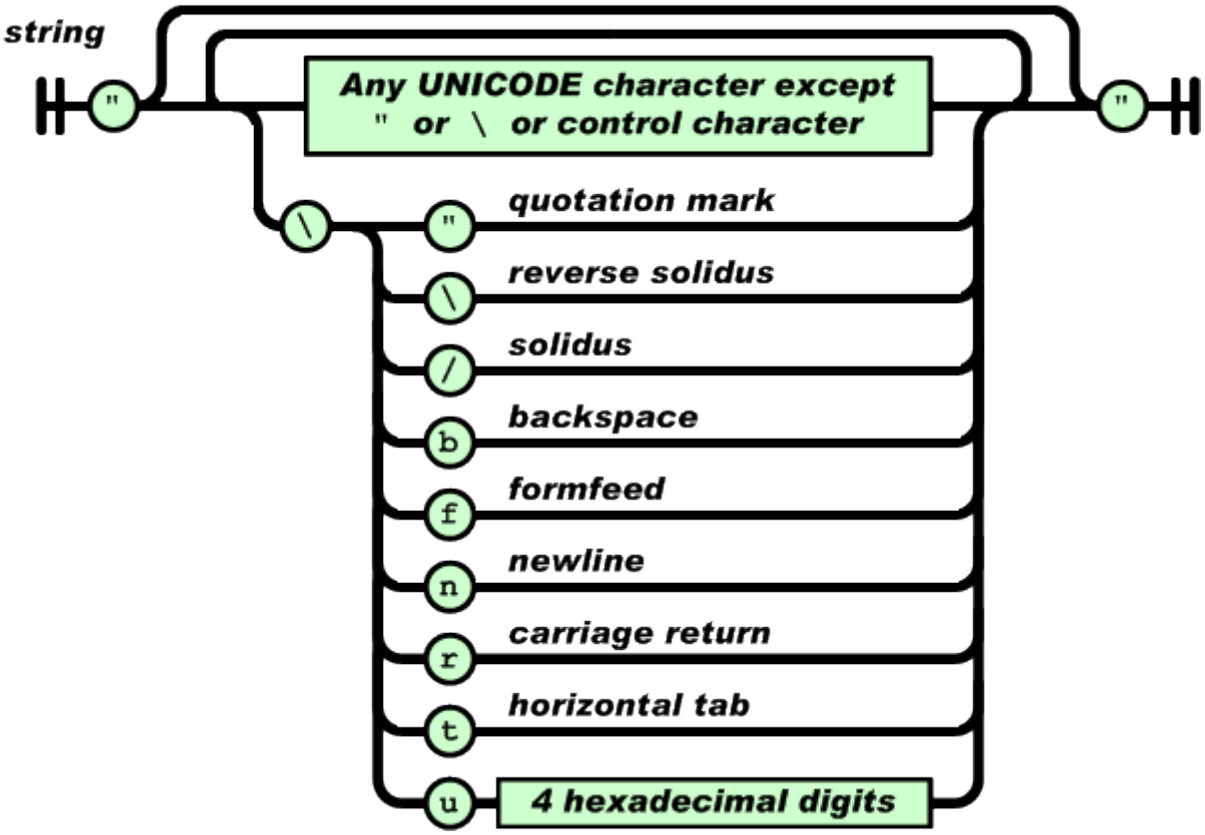


Wert - eine Zeichenkette, die in doppelte Anführungszeichen eingeschlossen ist, Zahl, wahr, falsch, Null, Objekt oder Matrix. Diese Strukturen können verschachtelt werden.



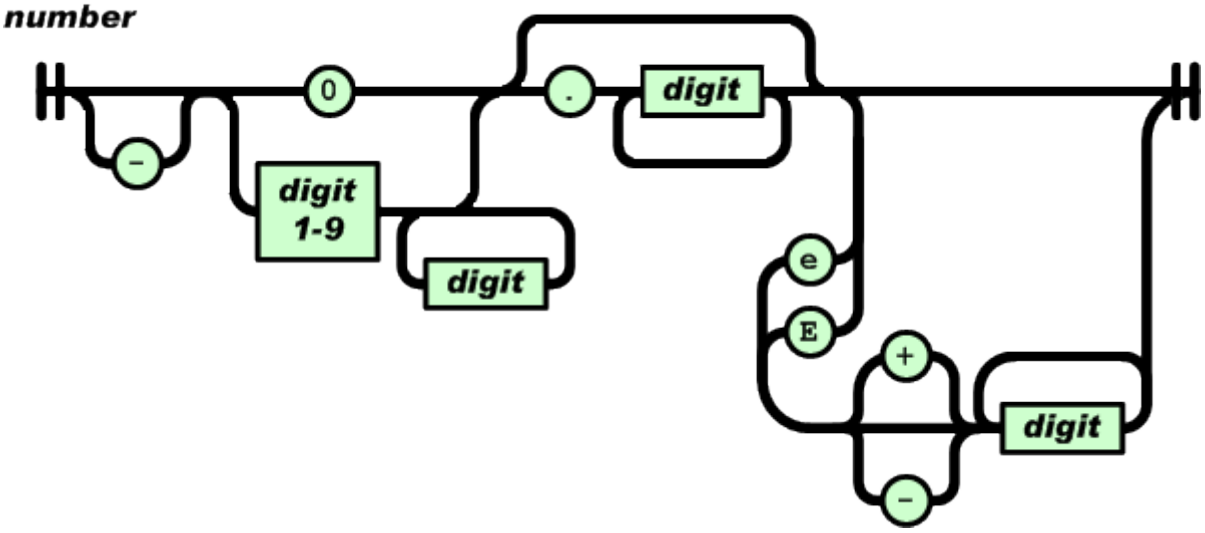
Zeichenkette - null oder mehr Unicode-Zeichen, eingeschlossen in doppelte Anführungszeichen und mit Escape-Sequenz unter Verwendung eines Backslash.

Zeichen - Zeichenkette mit einem einzelnen Zeichen



Eine Zahl - ähnlich wie C- und Java-Zahlen - verwendet keine oktalen oder hexadezima-

len Einträge.



7. Serverteile von Web-Technologien

Zuerst müssen wir definieren, was ein Webserver ist.

Als Webserver kann man einen Computer bezeichnen, der für die Bearbeitung von HTTP(S) Anfragen von Clients (meist Webbrowsern) zuständig ist. Unter Erfüllung der Anforderungen versteht man das Senden eines bestimmten URL-Ziels (typischerweise eine Webseite, aber auch einen statischen Text, ein Bild oder eine andere Datei). Webseiten sind in der Regel HTML-Dokumente. Oder ein Computerprogramm, das die oben beschriebenen Aktivitäten ausführt (ein Dämon).

HTTP ist ein Internet-Protokoll zum Austausch von Hypertext-Dokumenten im HTML-Format und verwendet den sogenannten Uniform Resource Locator (URL), der die eindeutige Position einer Quelle im Internet angibt. HTTP erlaubt keine Verschlüsselung oder Datenintegritätssicherheit, dafür dient HTTPS.

HTTPS ist ein Protokoll, das eine sichere Kommunikation in einem Computernetzwerk ermöglicht, HTTP wird zusammen mit SSL oder TLS verwendet.

HTTPS-Vorteile sind

- Authentifizierung
- Vertraulichkeit der übermittelten Daten
- Integrität des Inhalts
- die Möglichkeit, das HTTP/2-Protokoll zu verwenden
- Vorteile der Google-Suche

Nachteile:

- Leistungsabfall auf älterer Hardware
- die Notwendigkeit eines Zertifikats und seiner Verlängerung
- erlaubt es nicht, bestimmte URLs zu blockieren, nur um die gesamte Website zu blockieren.
- etwas kompliziertere Webserver-Konfiguration
- mögliche Komplikationen mit älteren Webbrowsern

Jeder Webserver ist mit einem Computernetzwerk verbunden und empfängt HTTP-Anfragen. Anforderungen werden behandelt und der Computer gibt eine Antwort zurück, normalerweise ein HTML-Dokument. (oder Text, Bild, etc.). Die Serverantwort erfolgt im HTTP-Format, wobei ein Header den Statuscode gefolgt vom Inhalt selbst enthält.

Antworten des Webservers (Statuscode)

- 2xx - erfolgreicher Abschluss der Anfrage
- 3xx - Probleme mit dem Umleiten
- 4xx - Fehler im Zusammenhang mit der Bearbeitung von Anfragen (Seite nicht verfügbar, etc.)
- 5xx - interne Serverfehler

Quelle der Serverinformationen kann sein:

- Statischer Inhalt - vorbereitete Datendateien (HTML-Seiten), deutlich schneller als dynamisch.
- Dynamischer Inhalt - Basierend auf der Anfrage werden Daten gesammelt (aus einer Datei, Datenbank oder einem Endgerät gelesen), formatiert und bereit für die Präsentation im HTML-Format und einem Webbrowser zur Verfügung gestellt.
- Dynamische Inhaltserstellung - Eine Vielzahl von Technologien (Perl, PHP, ASP, ASP.NET, JSP, Python, etc.) können viel mehr Informationen liefern und auf verschiedene "ad hoc"-Anfragen reagieren.

In der Praxis werden beide Ansätze kombiniert - Caching, node.js, ...

Bei Echtzeit-Traffic kann der Webserver überlastet sein.

Überlastungssymptome:

- langsame Serverreaktion (von Einheiten bis zu Hunderten)
- Fehler 500, 502, 503, 503, 504
- Die TCP-Verbindung ist gezwungen, neu zu starten, bevor die Antwort eintrifft.
- der Server sendet unvollständige Inhalte (dieses Verhalten wird meist durch einen Fehler verursacht).

Gründe für eine Überlastung:

- Klassische Überlastung (zu viele Leute sind gleichzeitig aktiv, aber nicht im Sinne eines Angriffs)
- DDoS-Angriff, Ein Computervirus, der viele Computer angreift und sie zwingt, sich zu verbinden.
- Internet-Boot
- Überlastung des physikalischen Netzwerks
- Die Inhalte sind auf mehrere Server verteilt und keiner von ihnen ist verfügbar. Alle Anfragen müssen von nur einem Server bedient werden.

Techniken zur Vermeidung von Überlastung:

- Netzwerkverkehrskontrolle mittels Firewalls, HTTP-Verkehrsmanagern und Traffic Shaping
- Verwendung von Web-Caches
- Verwendung verschiedener Domänennamen für statische und dynamische Abfragen
- Verwendung verschiedener Domänennamen und / oder Computer zur Trennung großer Dateien, so dass die kleinen im Cache gespeichert werden können.
- Verwendung mehrerer Webserver auf einem Computer, die jeweils mit einer benutzerdefinierten Netzwerkkarte ausgestattet sind.
- Verwendung mehrerer Computer, die miteinander verbunden sind und nach außen hin wie ein großer Server aussehen.
- Hinzufügen von mehr Hardware (RAM, CPU)
- Tuning der verwendeten Software

8. Serverteile von Webtechnologien II

Es gibt mehrere Arten von Webservern. Die häufigsten sind:

- Apache HTTP-Server
- IIS - Internet-Informationsdienst
- nginx
- GWS - Google Web Server

Apache HTTP-Server ist ein Open-Source-Software-Webserver für GNU / Linux, BSD, Solaris, Mac OS X, Microsoft Windows und andere Plattformen.

Es unterstützt eine Vielzahl von Funktionen - kompilierte Module, welche die wichtigsten Programmiersprachen auf der Serverseite erweitern (Perl, Python, Tcl, PHP....).), verschiedene Authentifizierungsschemata (mod_access, mod_auth, mod_digest und mod_auth_digest), Unterstützung für SSL, TLS (mod_ssl), ein Proxy (mod_proxy), URL-Rewriter, bekannt als Rewrite-Engine von mod_rewrite, Konfiguration von Protokolldateien (mod_log_config) und Filterung (mod_include und mod_ext_filter).

Enthält ein externes Datenkompressionsmodul (mod_gzip), ein Open-Source-Modul zum Schutz und zur Verhinderung von Webanwendungen vor Angriffen (mod_security).

Protokolle können mit Hilfe von Browsern und Skripten wie AWStats / W3Perl oder Visitors analysiert werden.

Unterstützung für viele grafische Umgebungen (GUIs)

Virtuelles Hosting - Eine Apache-Installation auf einem physischen Computer bedient mehrere Websites.

IIS - Internet Information Service ist ein Software-Webserver mit einer Sammlung von Erweiterungsmodulen, die von Microsoft für das Windows-Betriebssystem erstellt wurden.

Es unterstützt eine Reihe von Protokollen - HTTP, HTTPS, FTP, FTPS, SMTP und NNTP.

Module für IIS 7.5

- FTP Publishing Service - Veröffentlichen Sie Inhalte sicher auf IIS 7-Servern mit SSL-Authentifizierung und Datentransfer.
- Administrationspaket - Unterstützung für die Verwaltung von Benutzeroberflächenverwaltungsfunktionen in IIS 7, einschließlich ASP.NET-Berechtigungen, benutzerdefinierte Fehler, FastCGI-Konfiguration und Anforderungsfilterung.
- Application Request Routing - Bietet ein Proxy-Routing-Modul, das HTTP-

Anfragen an Content-Server weiterleitet, basierend auf dem HTTP-Header von Servervariablen und Ausrichtalgorithmen.

- Database Manager - Einfache Verwaltung lokaler und entfernter Datenbanken in IIS Manager.
- Mediendienste - Verbindet die Medienplattform mit dem IIS, um Multimedia- und andere Webinhalte instand zu halten und zu verwalten.
- URL-Rewrite-Modul - Bietet einen Umschreibmechanismus, der die URL-Anfrage ändert, bevor sie auf einem Webserver verarbeitet wird.
- WebDAV - Ermöglicht es Seitenautoren, Inhalte sicher auf IIS 7-Servern zu veröffentlichen.
- Web Deployment Tool - Synchronisiert IIS 6.0 und IIS 7 Server. Änderungen IIS 6.0 zu IIS 7

Nginx ist ein Software-Webserver mit Lastmanagement und Reverse Proxy mit Open Source Code. Es funktioniert mit den Protokollen HTTP (und HTTPS), SMTP, POP3, IMAP und SSL. Es konzentriert sich in erster Linie auf hohe Leistung und geringen Speicherbedarf. Es ist erweiterbar auf Unix, Linux und Unix-ähnliche Systeme unter BSD, es gibt Varianten für Solaris, MacOS und MS Windows.

Das grundlegende Ziel von Nginx ist die schnelle Verteilung von statischen Inhalten und die Möglichkeit, die Last auf andere Server entsprechend der eingestellten Priorität zu verteilen.
Das System ermöglicht es Ihnen, einen Backup-Server zu definieren, an den die Nginx-Anfrage weitergeleitet wird, es sei denn, der primäre Server reagiert auf eine bestimmte Grenze.

Eingehende Anfragen Nginx-Prozesse und asynchrone Prozesse

Ein eingehender HTTP- (oder HTTPS-) Request versucht zunächst, in seinem Cache zu suchen (er hat eine konfigurierbare Größe und Verweildauer), wenn er ihn findet, antwortet er sofort. Andernfalls wenden sie sich an einen der definierten Server (jeder Server hat eine definierte Priorität). Wenn der Server in der Lage ist, innerhalb einer bestimmten Zeit zu antworten, antwortet er/sie; andernfalls wendet er sich an den Backup-Server (natürlich, wenn definiert). Die Antwort, wenn möglich, wird in ihrem Cache gespeichert, und nachfolgende Abfragen zum Timeout des Lifetime-Cache werden im Cache behandelt.

Es existiert die Möglichkeit der Einstellung der Verbindungsbegrenzung von einer IP-Adresse aus.

Nginx ist ein modulares System.

Eines der Module - GEO Locations - ermöglicht es beispielsweise dem Land, Anfragen für definierte Server zu übermitteln oder den Zugriff auf Websites aus einigen Ländern zu

deaktivieren.
Das Modul leitet nach definierten Regeln weiter, Passwortsicherheit, gzip-Kompressionsunterstützung, Streaming (FLV, MP4).....

9. PHP: Hypertext Preprocessor

Programmierparadigma von PHP - zwingend, objektorientiert, prozedural, reflektierend.

Es wurde 1995 gegründet vom Autor Rasmus Lerdorf. Das erste Release war der 8. Juni 1995, die neueste Version ist 7.2.0 (30. November 2017).

PHP zeichnet sich durch eine schwache und dynamische Typkontrolle aus.

Die wichtigsten Implementierungen von PHP sind Zend Engine, Phalanger, Quercus, Project Zero, HipHop, etc.

PHP ist eine Programmiersprache, die speziell für die Programmierung dynamischer Websites und Webanwendungen entwickelt wurde. Die Skripte werden serverseitig ausgeführt - der Benutzer wird auf das Ergebnis seiner Aktivität übertragen. PHP-Skriptinterpreter können über die Befehlszeile, HTTP-Abfragen oder Webservices aufgerufen werden. Die Syntax der Sprache ist von mehreren Programmiersprachen (Perl, C, Pascal und Java) inspiriert.

Die PHP-Sprache ist plattformunabhängig, die Unterschiede zwischen den verschiedenen Betriebssystemen beschränken sich auf mehrere systemabhängige Funktionen, und Skripte können in der Regel ohne Änderungen zwischen den Betriebssystemen übertragen werden. Es unterstützt viele Bibliotheken für verschiedene Zwecke - z.B. Textverarbeitung, Grafik, Dateiverarbeitung, Zugriff auf die meisten Datenbanksysteme (z.B. MySQL, ODBC, Oracle, PostgreSQL, MSSQL), zählt zu den Internet-Protokollen (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP...).

PHP ist die am weitesten verbreitete Skriptsprache für das Web.

PHP-Eigenschaften

- Die PHP-Sprache ist dynamisch typisiert - der Datentyp der Variablen ist an einen Wert gebunden, nicht an eine Variable.
- Felder sind assoziativ
- Zeichenketten können in PHP auf 2 verschiedene Arten geschrieben werden:
    - um die Angebote zu zitieren (die Auswertung erfolgt durch das Ersetzen der Variablen innerhalb).
    - Schließen in Apostrophe (nur die Escape-Sequenz \ ' wird ersetzt).
- Variablen können erstellt und aufgehoben werden.
- Konstanten können definiert werden, können aber nicht gelöscht werden.
- Variablen haben ihre Sichtbarkeitsebenen und Regeln für ihre Persistenz.
- Unterstützt Referenzen, die verwendet werden können, um Referenzen auf andere Variablen oder Variablen im Feld zu speichern.
- Als Referenz können auch die Funktionsparameter aufgerufen werden - für jede

Variable zeichnet sie auf, wie viel sie von der Referenz referenziert wird, und entscheidet somit, wann sie die Variable abbrechen kann.

Vorteile:

- PHP ist für Websites gedacht.
- Umfangreicher Funktionsumfang in der PHP-Basisbibliothek (über fünfeinhalbtausend), zusätzliche Funktionen in PECL .
- Native Unterstützung für viele Datenbanksysteme.
- Multiplattform (insbesondere Linux und Microsoft Windows)
- Die Möglichkeit, native Betriebssystemfunktionen zu verwenden (möglicherweise Inkompatibilität mit einem anderen Betriebssystem).
- Strenge Lernkurve.
- Großer Support für Hosting-Dienste
- Eine Vielzahl von Projekten und Codes, die kostenlos verwendet werden können (WordPress, phpBBB und mehr).
- Relativ gute Dokumentation
- Sehr freie Lizenz

Nachteile:

- PHP-Sprache ist seit langem nur durch ihre Implementierung definiert, die offizielle Sprachspezifikation wurde Ende Juli 2014 veröffentlicht.
- Inkonsistente Benennung von Merkmalen
    - strpos (), strchr (), aber str_replace (), str_pad ().
- Uneinheitliche Nomenklatur der Funktionsgruppen
    - mysql_XXXXXX, imap_XXXXXX, json_XXXXXX (mit Unterstrich) versus imageXXXXXX, bcXXXXXX, gzXXXXXX (ohne Unterstrich).
- Uneinheitliche Reihenfolge der Parameter, z.B.: array_map () vs. array_filter ().
- Obwohl die Sprache Ausnahmen unterstützt, wird ihre Bibliothek selten verwendet.
- Schwächere Unicode-Unterstützung, nur über die PHP-Bibliothek (in Versionen nach PHP 5 sollte die Unicode-Zeichenkette der Grundtyp sein).
- In der Standardverteilung fehlt das Debugging (Debugging) Werkzeug.
- Nach der Bearbeitung der Anfrage pflegt sie den Kontext der Anwendung nicht, sie erstellt ihn immer wieder neu (schwächt die Performance).
- Die PHP-Sprache ist nicht nur für kleine Projekte und Seiten gedacht, sie kann auch in jede beliebige Anwendung programmiert werden.

Ausgewählte Großprojekte in PHP:

- MediaWiki - Software zur Erstellung von Wiki-Webprojekten,
- phpBBB - ein Paket für den Betrieb eines Webforums
- WordPress - ein Publikationssystem für Blogging und ähnliche Anwendungen

- Adminer - Webanwendung zur Verwaltung des MySQL-Datenbanksystems
- phpMyAdmin - Webanwendung zur Verwaltung des MySQL-Datenbanksystems
- Texy! - Intuitiver Syntax-Compiler zur Formatierung von Text in HTML
- Nette Framework - ein Framework zur Erstellung von Webanwendungen in PHP
- Facebook

10. PHP II - Syntax

PHP-Skript kann in HTML auf verschiedene Weise gekennzeichnet werden

- <? [PHP-Code] ?>
- <? php [PHP-Code] ?>
- <SCRIPT LANGUAGE = " php "> php [PHP code] </ SCRIPT>

Die einzelnen Anweisungen (Befehle) sind durch ein Semikolon getrennt. Den Kommentaren geht ein doppelter Schrägstrich oder ein Gitter voraus. Zwischen dem Schrägstrich werden mehrzeilige Kommentare geschrieben - Stern-Schrägstrich (//, #, / * mehrzeiliger Text * /)

Arten von Variablen in PHP

- Variablen
- Logischer Typ - Boolean - True / False, geschrieben als TRUE und FALSE (Größe spielt keine Rolle)
- der ganzzahlige Typ - Ganzzahl - eine positive oder negative Zahl (eine Null) von etwa -2 bis + 2 Billionen
- Dezimalzahl - Float, Real - mit Genauigkeit bis 14 Dezimalstellen
- Zeichenkette - String - Textketten

Der Variablentyp wird zum Zeitpunkt der Wertzuweisung bestimmt, während des Programms kann die Variable ihren Typ ändern, entweder durch Codebefehl oder durch eine Berechnung. Jede Variable muss einen eindeutigen Namen haben, beginnend mit dem Dollarzeichen ($) und ohne ein Leerzeichen gefolgt von einem Namen. Das erste Zeichen dieser Benennung muss entweder der Buchstabe a-z oder der Unterstrich sein. Es kann keine Zahl oder etwas anderes sein. Variablennamen unterscheiden zwischen Groß- und Kleinbuchstaben. Für die Zuordnung wird ein Zeichen gleich (=) verwendet.

PHP ermöglicht es Ihnen, drei Arten von Feldern zu definieren:

- Indexiert
- Assoziativ
- Mehrdimensional

Diese werden als Listen, Simulation eines Wörterbuchs und Sammlung von Elementen verwendet. Wir können mit Arrays als Stapel oder Warteschlangen arbeiten, und sie können auch Baumstrukturen darstellen (ein Feldelement kann ein Feld sein).

Felder können von PHP-Funktionen (Datenbank) zurückgegeben werden.
PHP gehört zu den PPE-Sprachen.

```
Class ClassName
{
  var $ VariableName
  function Function Name (parameters)
  {
        body function
  }
}
```

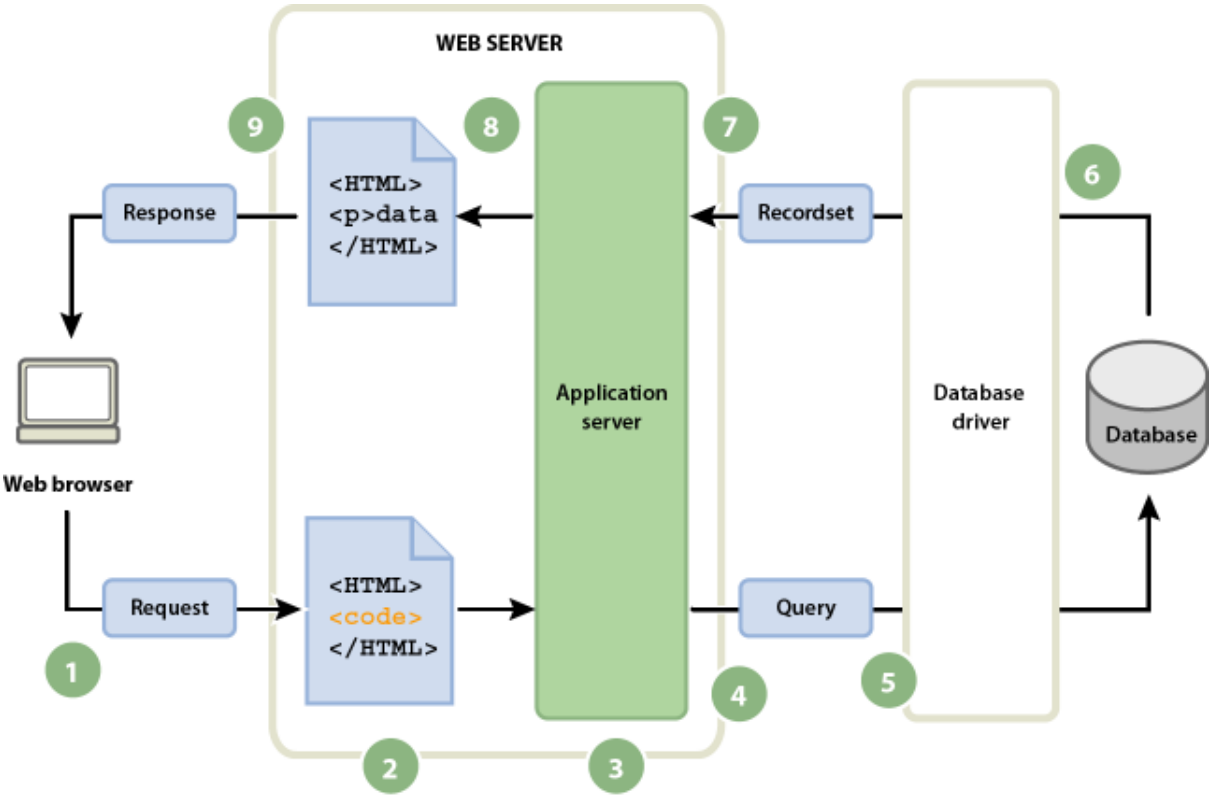
Vorsicht vor Vererbung - PHP hat keine privaten Methoden (Funktionen).

Darüber hinaus ermöglicht PHP die Eingabe und Änderung von NONDECLARED-Attributen!

Viele fertige Bibliotheken und Teile des PHP-Objektcodes werden verwendet, Sie können nur eine "Ebene" von Objekten ohne Vererbung haben. Ein Objekt kann ohne Methoden existieren, nur mit Attributen. Aber manchmal ist es besser, das Feld zu benutzen. Jemand in PHP, der jedes Attribut "zuweist", erstellt eine Methode oder mehrere Methoden. Bei der Verwendung von Objekten ist es mehr denn je wichtig, dass man über Konventionen nachdenkt und ihnen folgt (jemand beginnt mit Methoden, die nicht von einem Unterstrich aufgerufen werden sollten).

11. Webanwendungen, die mit Datenbanken arbeiten

Wie kann man eine Webanfrage in einer Datenbank bearbeiten?



1. Anfrage
2. HTML-Code
3. Anwendungsserver
4. Abfrage - Abfrage
5. Datenbanktreiber
6. Ausgabe aus der Datenbank
7. Recordset für den Server
8. In HTML übersetzen
9. Antwort

Die Grundlage für die Arbeit mit der Datenbank ist SQL.

SQL - Structured Query Language - mit einer kniffligen Sprache für die Arbeit mit relationalen Datenbanken.

Die Datenbank ist ein Dateisystem mit einer festen Datensatzstruktur, bei dem die Da-

teien über die Schlüssel miteinander verbunden sind.

Datenbank-Typen

- Hierarchische Datenbank
- Netzwerkdatenbank
- Relationale Datenbanken
- Objektdatenbank
- Objektbasierte relationale Datenbanken

Datenbankobjekte

- TABLE - ein einfaches Datenbankobjekt, mit dem Daten direkt im relationalen Datenbankspeicherplatz gespeichert werden können.

- VIEW - ein Datenbankobjekt, das dem Benutzer eine Vorschau auf die in der Tabelle enthaltenen Daten bietet.

- INDEX (KEY) - zur Beschleunigung von Such- und Abfrageprozessen, Definition eines eindeutigen Wertes mit Tabellenpeeling, Suchoptimierung

- CONSTRAINT - ermöglicht es Ihnen, Einschränkungen hinsichtlich der Bedingungen zu erstellen, die für die Werte seiner Spalten beim Einfügen oder Ändern von Datensätzen erfüllt sein müssen.

- TRANSAKTION - eine Gruppe von Befehlen, die die Datenbank von einem konsistenten Zustand in einen anderen konvertieren.

- TRIGGER - definiert die Aktionen, die bei einem definierten Ereignis oberhalb der Datenbanktabelle ausgeführt werden sollen.

- SQL-Datenverwaltungsbefehle (DML)

- SELECT - wählt Daten aus der Datenbank aus, ermöglicht die Auswahl von Unterbedingungen und die Sortierung der Daten.

- INSERT - fügt neue Daten in die Datenbank ein.

- UPDATE - Ändert die Datenbankdaten (Bearbeitung).

- MERGE - INSERT und UPDATE kombinieren die Daten (wenn es keinen entsprechenden Schlüssel gibt), falls sie existieren, und ändern sie dann im UPDATE-Stil.
- DELETE - entfernt Daten aus der Datenbank.

- EXPLAIN - ein spezieller Befehl, der das Verfahren zur Verarbeitung von SQL-Anweisungen anzeigt. Hilft dem Benutzer, Befehle zu optimieren, damit sie schneller sind.

- SHOW - ein weniger gängiger Befehl, mit dem Sie Datenbanken, Tabellen oder deren Definitionen anzeigen können.

- SQL-Befehle für die Datendefinitionssprache (DDL)

- CREATE - Erstellen neuer Objekte.

- ALTER - Änderungen an bestehenden Objekten.

- DROP - Entfernen von Objekten.

- SQL-Befehle für die Datenverwaltung (DCL)

- GRANT - ein Befehl zum Zuweisen von Berechtigungen an den Benutzer zu bestimmten Objekten.

- REVOKE - Befehl zum Entfernen für den Benutzer.

- START TRANSACTION - startet die Transaktion.

- COMMIT - Transaktionsbestätigung.

- ROLLBACK - Bricht die Transaktion ab, kehrt in den Ausgangszustand zurück.

Schlüsselwörter SQL für die Abfrage

- TOP - gibt die ersten N Zeilen zurück

- LIMIT - Begrenzt die Anzahl der vom SELECT-Befehl zurückgegebenen Zeilen.

- JOIN (FULL LEFT RIGHT INNER CROSS) ON - kombiniert das Ergebnis einer SELECT-Abfrage aus zwei Eingabesätzen (typischerweise eine Tabelle zu einer relationalen Datenbank).

- UNION - Vereinheitlichung des Abfrageergebnisses aus zwei oder mehr SELECT-Abfrageeingabesätzen

- ORDER BY - sortiert die mit der SELECT-Anweisung ausgewählten Einträge.

- WHERE - schränkt die Auswahl von Zeilen aus Tabellen über Bedingungen ein.

- GROUP BY - die Aggregation von Datensätzen, die durch die SELECT-Anweisung ausgewählt wurden.

- WITH ROLLUP - bei einem Standard-Dump erscheint anstelle einer Spalte eine Zeile mit einem NULL-Wert, nach der die Daten aggregiert werden (falls angegeben).

- HAVING - ermöglicht es Ihnen, die Zeilen einzuschränken, die von der Aggregationsfunktion verarbeitet werden.

Datenbanken und Datenbankserver

- Microsoft Access
- MySQL
- Orakel
- Microsoft SQL Server
- SQLite

12. DOM - Dokumentenobjektmodell

DOM ist das Document Object Model, API (Application Programming Interface), das einen gemeinsamen Standard für den Zugriff auf ein gültiges HTML-Dokument definiert oder ein richtig strukturiertes XML-Dokument ist völlig unabhängig von der Programmiersprache.

Mit DOM ist es möglich, einzelne Elemente (Objekte) über JavaScript zu behandeln.

DOM-Definitionen beschreiben einzelne Ebenen

- Ebene 0
  Zwischen-DOM-Unterstützung, welche vor der Erstellung von DOM Level 1 bestand. Zum Beispiel das von Microsoft entwickelte DHTML-Objektmodell oder das unbenannte Zwischen-DOM von Netscape. Stufe 0 ist keine vom W3C veröffentlichte formale Spezifikation, sondern wird als verständliche Abkürzung verwendet, die sich auf Dinge bezieht, die vor dem Standardisierungsprozess existierten.
- Ebene 1
  Navigieren Sie im DOM (HTML und XML) des Dokuments (oder seiner Baumstruktur) und manipulieren Sie den Inhalt (einschließlich dem Hinzufügen von Elementen). Spezifische HTML-Elemente sind ebenfalls enthalten.

- Ebene 2
  Unterstützung für Namensräume, Ereignisse und gefilterte Ansichten.

- Ebene 3
  Standardisierter Lade- und Speichermechanismus und Unterstützung von XML-Schemas. Ermöglicht das dynamische Einfügen von Inhalten in ein Dokument und fügt neue Methoden und Eigenschaften hinzu.

- Ebene 4
  Zusammenführung der bisherigen Standards DOM Level 3 Core, Element Traversal, Selectors API Level 2 DOM Level 3 Events und DOM Level 2 Traversal und Range and Simplification Ansatz und bestehende Standards, insbesondere Spezifikationen JavaScript und HTML5. Die Spezifikationen vereinfachen auch die häufigen DOM-Operationen.

Grundgedanken von DOM sind, alle HTML-Elemente als Objekte zu betrachten. Dann hat jedes Objekt Eigenschaften - Attribute - und kann auf Ereignisse reagieren.

Jedes Objekt muss identifiziert werden; es ist notwendig, um die Eigenschaften oder den Inhalt mit Hilfe eines Skripts zu ändern. Die Identifizierung erfolgt über ID oder Name.

DOM - Universelle Eigenschaften und Methoden zum Durchschauen und Lesen (Ver-

wendung: Dokument.Name)

- documentElement - gibt das root ("root") Dokumentelement zurück.
- getElementsByTagName () - gibt das Feld aller Elemente des angegebenen Namens zurück.
- parentNode - gibt das übergeordnete Objekt des Objekts zurück.
- nextSibling - gibt das nächste Geschwisterpaar des aktuellen Objekts zurück, wenn es existiert, andernfalls gibt es null zurück.
- previousSibling - gibt das vorherige Geschwisterpaar des aktuellen Objekts zurück, wenn es existiert.
- firstChild - Gibt das erste Kind des Objekts.
- lastChild - Gibt das letzte Kind des Objekts zurück.
- childNodes [] - gibt das Array aller Objekte ("Nodelists") zurück, die die Kinder des Objekts sind.
- nodeName - gibt den Namen des Objekts zurück.
- nodeValue - Liefert den Wert (Inhalt) des Objekts.
- data - gibt den Wert (Inhalt) des Textobjekts zurück.
- className - gibt den Wert des Klassenattributs zurück (nur HTML)
- id - gibt den Wert des Attributs der Objekt-ID zurück, das ein Elementtyp sein muss, nur für HTML.
- title - gibt den Wert des title-Attributs des Elements zurück, nur HTML
- item () - spezifiziert ein spezifiziertes Objektfeld ("Nodelist"), das durch den Index spezifiziert wird.

DOM - Methoden zur Manipulation von Knoten

- createElement (name) - erstellt ein neues Element
- setAttribute (Name, Wert) - Einstellungen des Attributs
- createTextNode (value) - Legt einen Textknoten an.
- splitText ("Division") - Aufteilen des Textknotens in Knoten zwei
- normalize () - Zusammenführen von Schwesterknoten vom Typ Text zu einem Knoten
- appendChild (object) - Fügt einen Kindknoten hinzu.
- insertBefore (object, object) - 2 Parameter - der erste ist der einzufügende Knoten und der andere ist der Knoten, in den wir einfügen werden.
- cloneNode (true | false) - erstellt eine Kopie des Objekts.
- replaceChild (object, object) - 2 Parameter - der erste Parameter ist ein Knoten, der den als zweiten Parameter angegebenen Knoten ersetzt.
- removeChild (Object) - Gilt für den übergeordneten Knoten des zu entfernenden Knotens.
- removeAttribute (name) - Entfernt den Attributknoten.
- DOM kann auch verwendet werden, um effizient dynamische Tabellen zu erstellen oder die Seitenformatierung mittels CSS dynamisch zu ändern.

INFORMATICS - ENGLISH

Interreg
EUROPEAN UNION
Austria-Czech Republic
European Regional Development Fund

INTRODUCTION

The presented technical book “study material for the field of informatics” as prepared within the project “Methodological Concept for Effective Support of Key Professional Competencies Using a Foreign Language - CLIL as a Teaching Strategy at a University” implemented with the financial support of the European Union programme INTERREG VA Austria - Czech Republic 2014 - 2020.

The project was realized in cooperation of two technically oriented higher education institutions, the Institute of Technology and Business in České Budějovice, the Czech Republic, and the University of Applied Sciences, Upper Austria. One of the main project outputs was the preparation of professional didactic materials for four technical disciplines (Informatics, Logistics and Transport, Mechanical Engineering, and Civil Engineering) taught at the partner institutions in three languages: Czech, German and English. As a teaching method, CLIL (Content and Language Integrated Learning) was chosen, as it combines teaching of content and a foreign language. Thus, the prepared materials are of great importance not only as teaching and learning material used by teachers and students at the above-mentioned technical universities, but they can also be used by experts in specific fields and employees of companies operating in the cross-border region, who thus have the opportunity to improve their professional language skills.

Teachers from both partner institutions and practitioners from both border regions participated in the preparation of the materials. Materials in the field of Informatics were prepared by teachers of content subjects at both partner universities. Their topics were selected and consulted in cooperation with practitioners. A total of twelve topics were prepared for this purpose: Computer Science, Algorithms and Data Structures, Databases, Information and Communication Technologies, Introduction to JAVA Programming, Object-Oriented JAVA Programming, Introduction to Media Studies, IT Security, Networks, Software Engineering and Modelling, Web Application Basics, Web technologies. The topics were chosen to meet the needs of practice and to cover the widest possible range, from the basics and theory to specific issues. Moreover, each of the topics is divided into sub-chapters so that it is possible to study the module as a whole or to choose only some chapters to study. The materials prepared are available online, allowing each student and teacher to compile the course or teaching content according to their specific needs.

As mentioned above, the materials have been prepared in three languages. Each topic prepared by content teachers was subsequently processed by linguistic experts in order to comply with the principles of the CLIL method and to acquire not only professional but also language skills. Currently, knowledge of a foreign language appears to be crucial for finding a suitable job. This publication can thus serve not only for content teachers and university students, but also for graduates and employers and employees of companies operating in the above-mentioned disciplines both in and outside the cross-border region, which represents its considerable added value.

COMPUTER SCIENCE

1. Basic mathematical concepts and numerical systems

- Data or information we can capture with our senses.
- Information is data, which we understand, make sense and can be quantified (measured) after conversion into numbers.
- Information can be: Text, Image, or Sound.
- Informatics is a science of preservation, processing by using data and information. It is Part I of mathematics and as a means to use computer technology.
- IT branches:
    - Computing Technology - Examines hardware
    - Algorithmization - design of problem solving procedures
    - Programming - converting algorithms into programming language
    - Software Engineering - Application Development
    - Computer Graphics - Learn about creating and processing 2D and 3D images
    - Computer modeling and simulation - Application of mathematical models to real-world situations
    - Formal logic, automata theory and formal languages - mathematical models of machines and formal languages for writing algorithms and programs
    - Cybernetics and robotics - development of machines capable of independent activities
    - Artificial Intelligence - Exploring the processes of human thinking, their mathematical description and application in machine development
- Hardware or computer hardware, a generic name for all the physical devices your computer is equipped with
- Software - It can be divided into a system and application.
- Information stored in we are measuring computers in Bit - b - and Byte - B units
- The information stored on your computer is measured in Bit - b - and Byte - B
- The bit is the base and the smallest unit of information has two values: 0, 1
- Byte as a unit of information has an 8-bit value.

Since informatics is part of mathematics, it also uses some of its terms, such as:

- Cartesian product
- Relation

- Morphism
- Divisibility
- The largest common divisor, the smallest common multiple
- Prime Numbers

Because the computer stores information in bits and multiples, it works in a binary numeric system. You need to say something about numerical systems.

Numerical systems are divided into positional and non-positional.

Position Numerical Systems - The value of each digit is given by its position in the symbol sequence

Their advantage is high elasticity and a relatively small set of digits; the disadvantage is a very easy change of the value of a number by simply adding a digit to the original number.

The key characteristic is the Base, then the weights of the individual digits are the basis of the base.

Examples:

Unary – the base 1

Binary - the base 2, two states - yes / no, 1 bit, the symbols 0, 1,

Octal - the base 8, one byte (= 8 bits), the symbols 0, 1, 2, 3, 4, 5, 6, 7

Decimal - the base 10, natural for people (ten fingers), the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (the number)

Hexadecimal – the base: 16, 2 bytes (= 16 bits), symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Sexagesimal – the base 60, time measurement

Non-Position Numeric Systems - A method of representing numbers in which the value of a digit is not given by its location in a given sequence of digits. These ways of writing numbers are hardly used today and are considered obsolete, yet they have certain advantages such as simple addition and subtraction. The drawbacks are that they often did not contain a symbol for zero and negative numbers, as well as a long count of numbers that significantly exceed the value of the largest symbol of the system

Example: Roman numerals

Transfer between systems

From a higher base system

Converting from a higher base system to a lower base system then we usually use a partitioning algorithm.

73/2 = 36 + 1 - 1 at the 1st place from behind
36/2 = 18 + 0 - 0 at 2nd place from behind
18/2 = 9 + 0 - 0 at 3rd place from behind

9/2 = 4 +1 - 1 at the 4th position from behind
4/2 = 2 + 0 - 0 at the 5th place from behind
2/2 = 1 + 0 - 0 to the 6th position from behind
1/2 = 0 +1 - 1 at the 7th position from behind

From a lower base system

Transferring from a lower base system to a higher base system is easier. We know that the digit at the n-th place (calculated from 0) represents the number resulting from the amplification of the base on the n-th. These numbers are numbered and transferred.

$$1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 = 73_{10}$$

2. Logic

Logic is the science of correct reasoning, or the art of proper argumentation.

What is the judgment (argument)?

Based on the truth of the premise (assumptions) P1 ... Pn it can be concluded that the conclusion Z is true

We recognize several types of argumentation logic.

Deduction

Conclusion Z logically follows from the assumptions $1, \ldots, Pn$, we denote $P1, \ldots, Pn \mid = Z$, if under no circumstances can the case arise such that the assumptions would be true and the conclusion would be untrue.

Example:

P1: All rabbits out of the hat are white.

P2: These rabbits are out of the hat

Z: ⇒ These rabbits are white.

Induction

Generalization - from specific to general

Example:

P1: These rabbits are out of the hat

P2: These rabbits are white.

Z: ⇒ (probably) All rabbits in the hat are white.

Conclusion Z only applies with a certain probability

Abduction

We create hypotheses for observed phenomena, diagnosis of "disorders"

Example:

P1: All rabbits out of the hat are white.

P2: These rabbits are white.

Z: ⇒ (probably) These rabbits are out of the hat

Conclusion Z only applies with a certain probability

Examples:

P1: All green toadstools are violent poisonous

P2: This fungus is a green toadstool.

Z: This fungus is violently poisonous.

Valid deduction

P1: All green toadstools are violently poisonous

P2: This pencil is green a green toadstool.

Z: This pencil is violently poisonous.

Correct Judgment, Conclusion False ⇒ At least one premise is untrue

Judgment has the correct logical form

Logical connectives - "and", "or", "if", "then", and others have fixed meanings, interpret them using elementary statements or parts thereof (predicates and functional terms).

Logic is a tool that helps to discover the relationship of logical outcome, to solve tasks such as "What does this imply?" It helps our intuition, which can sometimes fail, because premise can be complexly formulated, intertwined and negated, the relationship of occurrence is not obvious at first view.

Example:

P1:	All men like football and beer
P2:	Some beer lovers do not like football
P3:	Xaver likes only lovers of football and beer
Z:	Xaver does not like some women.

Essentially, if all assumptions are true, then the conclusion must be true.

Is Judgment Valid?

Of course, if Xaver likes only lovers of football and beer (3.), he does not like some beer lovers (those who do not like football (2.)), he does not like (according to 1.) some "non-men" i.e. women.

However, it is not valid under the definition

Judgment is valid if necessary, i.e. under all circumstances (interpretations) when true assumptions are true and true.

But: in our example, those individuals who are not men would not have to be interpreted as women. There is no premise here, "who is not a man, is a woman," likewise we still need the premise "who is a lover of something he likes".

We must put all the assumptions necessary to draw the conclusion

P1: All men like football and beer.

P2: Some beer lovers do not like football.

P3: Xaver likes only lovers of football and beer.

P4: Who is not a man is a woman.

P5: Who's a lover of something, he likes it.

Z: Some women do not like Xaver.

Now the judgment is correct, it has a valid logical form.

Conclusion logically follows from the assumptions (they are "informally, deductively included").

Logic properties

Valid (correct) judgment may have a false conclusion

Usage: Proof AD ABSURDUM

Monotony

If judgment is valid, extending the set of assumptions to a further assumption does not lead to a change in the validity of the judgment

From the controversial assumptions, any conclusion is drawn

Reflexivity, Transition

Valid Judgment Schemes

1. $A \supset B, A| = B$

   modus ponens

   P1: No prime number divisible 3

   P2: 9 is divisible 3

   Z: 9 is not a prime number

2. $A \supset \neg B, B| = \neg A$

   modus ponens + transposition

   P1: All people are reasonable

   P2: The stone is not reasonable

   Z: This stone is not human

3. $A \supset B, \neg B| = \neg A$

   modus ponens + transposition

   P1: If 12 is prime or not divisible 3

   P2: 12 is divisible 3

   Z: 12 is not a prime number

4. $\neg A \vee \neg B, B| = \neg A$

   elimination of disjunction

   P1: 12 is not a prime number or is not divisible 3

   P2: 12 is divisible 3

   Z: 12 is not a prime number

3. Sets and Relations

3.1. Sets

Definition - a summary of objects that are precisely defined and identifiable and forms part of the world our ideas and thoughts; these objects are called elements of the set.

We use the following symbol for sets:

- Sets: A, B, C ...
- Elements : a, b, c ...
- Element Sets : $a \in A$
- For all x applies P : ∀x : P
    - There is x for which P holds: ∃x : P
    - There is just one x such that P holds : ∃!x : P
- Conjunctions, disjunctions, negations: ∧, ∨, ¬
- Implications, Equivalences: ⇒ , ⇔
- Sum, multiplication: Σ, Π

The set can be made

- Enumeration of elements : $A = \{1,2,3,4,5\}$
- Using the characteristic features: $A = \{a \in N | a < 6\}$

We define empty set: $\{\emptyset\}$, which contains no elements.

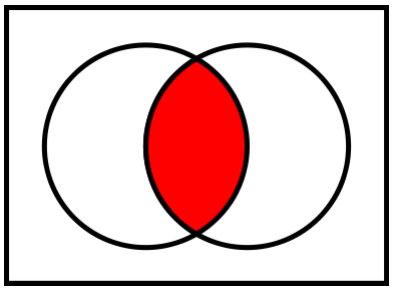
The number of element elements determines - **cardinality** - | A |, where A is a set.

We recognize sets:

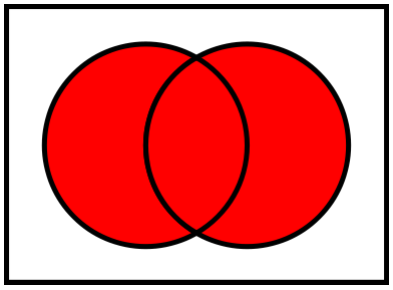
- Finite
- Infinite
- countable
- Continuum

Basic Operations with sets:

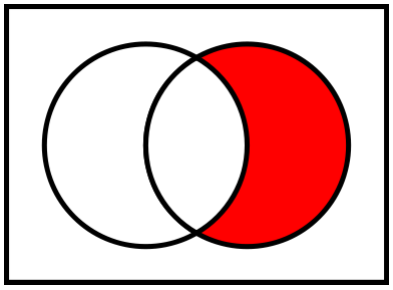
Intersection: $A \cap B$



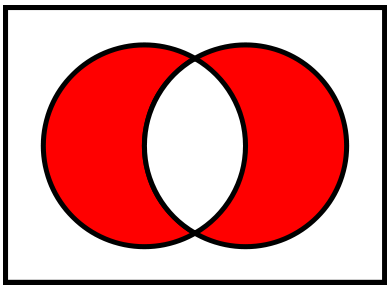
Union: $A \cup B$



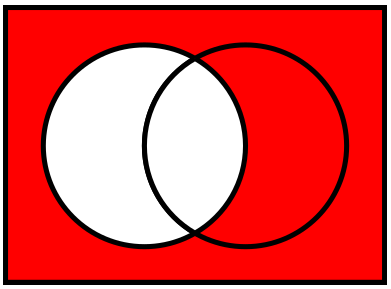
Difference of sets: $B \setminus A$



Symmetrical difference: $A \triangle B$



Complement of A in U: $A^c = U \setminus A$



Subset (inclusion– the opposite of exclusion): $A \subseteq B$

$$(\forall x)(x \in A \Rightarrow x \in B)$$

Potential set - set of all subsets:

$$P(\emptyset) = \emptyset$$

$$P(\{a\}) = \{\emptyset, \{a\}\}$$

$$P(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$$

3.2. Relations

Definition: n-ary relation between sets $A_1, A_2, A_3, \ldots, A_n$ where $n \in N$, we mean any subset of the Cartesian product *of n* sets.

Cartesian product (discrete product)

Set $X \times Y$, which contains all the ordered pairs, where the first entry is from X and the second entry is z Y

A\B	1	2	3
X	(X,1)	(X,2)	(X,3)
Y	(Y,1)	(Y,2)	(Y,3)
Z	(Z,1)	(Z,2)	(Z,3)

Relations can be classified by arity to:

Unary - each subset of the set N

Example: a positive number is a true / false statement

Binary - Each set of ordered pairs $[x; y] \in M^2$

Eg: is greater than, is smaller than, is a subset

Ternary - Each set of ordered triplets $[x; y; z] \in M^3$.

Example: lies between

N- aray - each set of ordered n-tuplets of M^n.

Operations with relations

In addition to classical set operations (all sessions must have the same arithmetic), we introduce the inverse relation as: R^{-1}: $\forall$ a $\in$ M_1 $\forall$ b $\in$ M_2:[a,b] $\in$ R $\Leftrightarrow$ [b,a] $\in$ R^{-1} and a compound session.

The binary relation on the set is:

- **reflexive**, if for all a ∈ M holds that [a, a] ∈ R.
- **Symmetrical** if for all a, b ∈ M applies if there is [a, b] ∈ R then [b, a] ∈ R.
- **Antisymmetric** if for all a, b ∈ M holds that if it is [a, b] ∈ R and also [b, a] ∈ R, then a = b.
- **Transitive** if for all a, b, c ∈ M holds that if [a, b] ∈ R while [b, c] ∈ R then [a, c] is in R.

Two special relations

- **Equivalence** is a relation that is reflexive, symmetric and transitive.
- **Order** is a relation that is reflexive, antisymmetric and transitive.

4. Relational structures

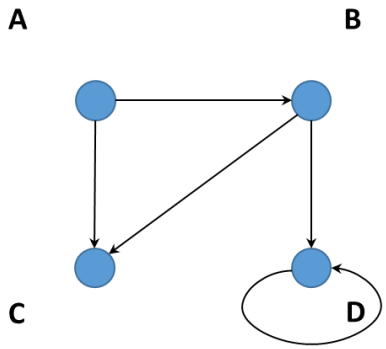
Relational structure means a mathematical structure that, in addition to the carrier set, contains one or more relations (which may have different and different arithmetic). These include oriented graphs and ordered sets (including their special cases, lines arranged linearly or well).

Directed graphs

A directed graph G is a pair (V, E), where E is a subset of the Cartesian product V × V × V $E \subseteq V$

Elements E are called arrows or oriented edges. Oriented edge e has the form (x, y). We say that this oriented edge is based on x and ends in y.

Directed graphs are represented by adjacency matrix, or list of vertices and edges



Adjacent Matrix :

		WHERE			
		A	B	C	D
FROM	A	0	1	1	0
	B	0	0	1	1
	C	0	0	0	0
	D	0	0	0	1

For directed graphs may not be symmetric, 1where is the edge, 0 where edge is missing

The set of vertices and edges G: (A, [A, B]), (A, [A, C]) (B, [B, C] D, D])

Ordered sets

Linear arrangement (= total ordering) is defined so that:

Each two elements of the linearly ordered set are comparable, ie there is a relation R on the set *X* and *a, b, c* ∈ *X* such that:

Transitivity: $(\forall x, y, z \in A)\big((xRy \wedge yRz) \Rightarrow xRz\big)$

Weak antisymmetry : $(\forall x, y \in A)\big((xRy \wedge yRx) \Rightarrow x = y\big)$

Trichotomy: $aRb \vee bRa \vee a = b$

Example: arrangement on a set of natural and real numbers

A well-arrangment is defined as:

The set S is well-arranged just when each non-empty part of the ordered set S has the smallest element

Example: Natural numbers

An ordered set is one on which an arrangement is defined.

Arrangement is a binary relation *R*, which is reflexive, transitive and weakly antisymmetric

$(\forall x \in A)(xRx)$ reflexivity - each element is in a *R* - relation to itself

$(\forall x, y, z \in A)\big((xRy \wedge yRz) \Rightarrow xRz\big)$ transitivity - if the element of the set is in the arrangement between two other elements, the two are also comparable

$(\forall x, y \in A)\big((xRy \wedge yRx) \Rightarrow x = y\big)$ weak antisymmetry (no cycles in arrangement)

= **fuzzy arrangement**

Sharp Arrangement - Reflexivity replaced by antireflexivity - no element is in relation with itself $(\forall x \in A)\big(\neg(xRx)\big)$

5. Projection

A special example of a binary relation that ensures unambiguous image to each pattern

If this is a binary session on a numeric set, then we call it as a function.

Definition:

Morphism f from the set A to the set B is such a binary relation for which each element x of A assigns no more than one element y of B so that $[x, y] \in f$

Designation:

$y = f(x) \quad f: x \to y$

Important notation: $y = f(x) \in B,\ x \in A$

y - The target of element x in morphism f, morphism value f at point x

x - Source of element y in view f

A set of elements $x \in A$, for which there is just one such element $y \in B$, that $y = f(x)$, is called the domain of f

B set of elements $y \in B$, for which there is at least one such element $x \in A$, that $f(x) = y$ is called the range of f

We distinguish several basic types of morphisms:

Morphism in Set (Morphism on Set):

$$f: A \to B,\ A = B$$

Set into a set - the whole set A is domain

$$f: A \to B,\ D(f) = A$$

From set to set (surjection) - the whole set B is a range of values

$$f: A \to B,\ H(f) = B, \forall b \in B: \exists a \in A: f(a) = b$$

Set on set

$$f: A \to B,\ D(f) = A \wedge H(f) = B,$$
$$\forall a \in A: \exists! b \in B: b = f(a) \wedge \forall b' \in B: \exists! a' \in A: f(a') = b'$$

Injection

$$\forall b \in B: \exists! a \in A: f(a) = b$$

Unambiguous (bijection) – Injection function of set A on set B (is injectable and surjective at the same time)

$$f: A \leftrightarrow B$$

Inverse - There are defined only for injection functions

If it is $f: A \to B$ injection, then morphism

$f^{-1}: B \to A$, which $\forall b \in H(f)$: $f^{-1}(b) = a \in D(f): y = f(x)$, is called inverse

$$D(f^{-1}) = H(f),\ f^{-1}(y) = x \Leftrightarrow f(x) = y$$

Morphisms can also be composed:

Sets A, B, C

Morphism: $f: A \to B$ and $g: B \to C$

Composed morphism: $h: A \to C,\ h = g \circ f,\ h(a) = g\big(f(a)\big)\ \forall a \in A$

It is generally not commutative, but is associative

Inversion of a composed morphism:

$$(g \circ f)^{-1} = f^{-1} \circ g^{-1}$$

6. Boolean algebra

Boolean algebra is defined as: Sixth $(A, \wedge, \vee, -, 0, 1)$, where A is non-empty set, $0 \in A$ - the smallest element , $1 \in A$ - the largest element , – unary operation (complement), ∧ -intersection, ∧ logical product , ∨ - logical sum .

It is based on axioms:

Commutativity

$$x \vee y = y \vee x$$
$$x \wedge y = y \wedge x$$

Distributivity

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Neutrality 0 and 1

$$x \vee 0 = x$$
$$x \wedge 1 = x$$

Complementarity

$$x \vee -x = 1$$
$$x \wedge -x = 0$$

A has the following characteristics:

- Absorption: $x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$
- Aggression of zeros: $x \wedge 0 = 0$
- Aggression of one: $x \vee 1 = 1$
- idempotence : $x \vee x = x$, $x \wedge x = x$
- Absorption negation: $x \vee (-x \wedge y) = x \vee y$, $x \wedge (-x \vee y) = x \wedge y$
- Double negation: $-(-x) = x$
- De Morgan's laws: $-x \wedge -y = -(x \vee y)$, $-x \vee -y = -(x \wedge y)$
- *0* and *1* are complementary to one another: -0 = 1, -1 = 0

Basic one-input operations:

- ID (Identity)
- NOT (negate)

Two Input Basic Operations.

- OR (logical sum)
- AND (logical product)

Truth table for basic operations:

A	B	ID(A)	ID(B)	NOT(A)	NOT(B)	A OR B	A AND B
0	0	0	0	1	1	0	0
0	1	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	1	1	1	0	0	1	1

Derived two-input operations:

- NOR (Not OR – negation of sum)
- NAND (Not AND - negating the product)
- Implication ⇒ (if the assumption A is fulfilled, B results, if the expectation is not fulfilled, something results and therefore 1)
- Equivalence of EQ ⇔ (matching of inputs)
- XOR (Exclusive OR - Unique Inputs)

Truth table for derived two-input operations:

A	B	NOR	NAND	⇒	⇔	XOR
0	**0**	1	1	1	1	0
0	**1**	0	1	1	0	1
1	**0**	0	1	0	0	1
1	**1**	0	0	1	1	0

Let's take the expression: A (B + NOT (C))

You can write this expression using:

1. **Truth tables:**

i	A	B	C	A(B+NOT(C))
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

2. **Algebraic expression:**

AND (NOT (B) (NOT (C)) + AB (NOT (C)) + ABC

This is the sum of all expressions giving the result 1

3. **Set the state index so that the lines in the truth table are numbered 0** (see Column *i*)

A (B+NOT(C)) = {4, 6, 7}

To minimize the algebraic expression, Karnaugh's map can be used.

It is filled in so that the columns (rows) above which the variable is given are for it 1

The table shows the values in the ABC order

			B	
		C		
	000	001	011	010
A	100	101	111	110

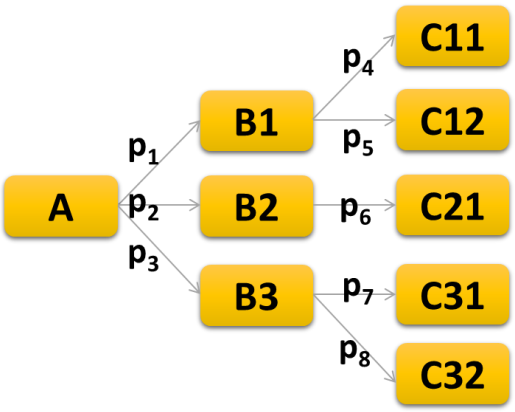
Example : A (NOT (B) (NOT (C)) + A (NOT (B)) C + ABC = A

			B	
		C		
	0	0	0	0
A	1	0	1	1

7. Probability and statistics

The difference between determinism and stochasticity is in that determinism is known in advance. FROM A to B goes with 100% probability (I'm always going to do that , another option is not).

In contrast, in the stochastic phenomena everything is just happening with with some probability, from And to B1 I'm going with probability p_1. But I can also go to B2 with probability p_2 and B3 s p_3. That is, I only know the probability with what happens the result. IN Random processes play a role in this phenomenon. If we look at the image, the sum of p1, p2 and p3 must be equal to 1, or z And I'm going to some B with 100% probability. Similarly, the probability p6 (B2-> C21) is equal to one .



Probability is defined as follows:

Let F be the experiment E, the set of conditions of the experiment Π and the set of all possible results F. F contains events 0, A, B... Ω, where 0 is an event that never occurs and Ω event that always occurs. Therefore, the experiment can be written as E = (Ω, F). The set of experiment results repeated under the same conditions is N, the individual results are 1 ... n and I is a subset of the results. Subset *A(I)* is subset of *I* such that the result of the experiment is an event *A.* The number of results in the subset *A (I)* is $n_A(I)$ and the set of all results is $\Omega(I)$. Then all subsets *I* apply , where *P (A)* is the probability of the phenomenon *A*:

$$P(A) \approx \frac{n_A(I)}{n_\Omega(I)}$$

The relative frequency of the *A* phenomenon can be defined as : $\frac{n_A(I)}{n_\Omega(I)}$

Probability properties - the probability of A is always between 0 and 1 including both extreme values.

The probability that phenomenon A does not occur, or negation of phenomenon A is 1-P (A).

If phenomena A and B do not occur simultaneously, then P $(A \cap B) = \emptyset$

Probability of occurrence of phenomenon *A* or phenomenon *B:* $P(A \cup B)$ is $P(A \cup B) = P(A) + P(B)$

Conditional probability - probability of A phenomenon when phenomenon B curs: $(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}$

$$\boldsymbol{(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}}$$

Bayes theorem: $P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$

Statistics is a set of concepts, rules and procedures that help us organize numerical information in the background to understand the techniques and make informed decisions.

Statistical knowledge is applied to the data, which is the facts and information from observations (ideally from experiments)

Data are divided into numerical (quantitative) data, which are the result of the measurement, and categorical (qualitative) data, divided into groups based on common characteristics.

The basic statistical variables are the mean value (weighted average), the median, which shows the central tendency, the middle value in the given set, and the modus as the most common value. Next, we calculate the variance, and from it the standard deviation as the root. We can also find covariance that determines how much the two values change together. And then skewness and kurtosis, revealing the distribution of values for a given file.

The distribution of file values describes the so-called distribution. One of the basic distributions is Gaussian (normal) distribution. It is symmetrical, having the same mean, mode, median, skewness and kurtosis are zero.

One of the basic statistics is the Central Limit Theorem, which expresses the fact that the sum of many independent random variables will tend to disperse in a small set of distribution functions.

8. Theory of Information

Information theory deals with the measurement, transmission, encryption, storage and subsequent processing of information in quantitative terms. Its founder is C.E. Shannon.

Information

What we exchange with the outside world when we adapt to it and adapt to it by adapting it

Content of the message, communication, clarification, explanation, instruction.

Data, numbers, characters, commands, instructions, commands, messages and the like. For information, consider also the ideas and perceptions received and emitted by living organisms.

The size information is provided orderliness (uncertainty) of the elements (system).

The system information is the greater the likelihood of occurrence of individual states is smaller. The information is bigger if the message contains something new that was not known or could not be easily guessed (it is unlikely)

For passing the information is needed

Any method of encoding = transfer to appropriate signals or symbols

Signal - information-bearing physical quantity

Message - Expression of information using a sequence of symbols (characters)

The report as such has three parts:

- Syntax - track
    - Syntactic content
    - It can exist itself without semantic and pragmatic content
    - It refers to the mutual arrangement of characters as carriers of information
    - Describes the quantitative information page
- Semantics - Information Content
    - Semantic content
    - The meaning of the message can not be measured
    - A message with the same amount of information can be written in different languages
    - Describes qualitative information page
- Importance - pragmatic content
    - Determines the significance (usefulness, value) of the message and priority
    - Qualitative information page

The information content of the message is measured by entropy.

Information is a measure of the amount of uncertainty or uncertainty about any random action eliminated by the realization of this happening.

We determine the degree of uncertainty as the entropy of the system

$$H(X) = -\sum_{i=1}^{n} p(x_i)p(x_i)$$

Information about System X can be obtained from:

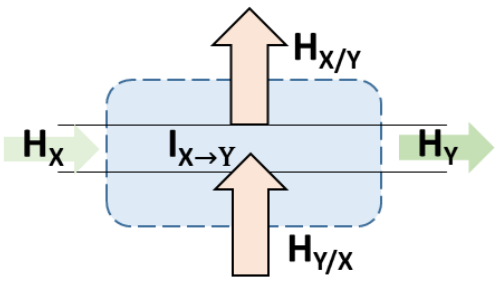
Direct observation

Indirectly through a system of Y (X system is unavailable to us)

System Status Y is not necessarily identical with the state of X (the text of the telegram in one city X, Y in the second city)

Differences of two kinds

- Some states of the X system appear in one state Y (Y can not distinguish the fines in X, Y is coarser than X)
- Errors in transmission between X and Y - eg: noise
- Using a channel



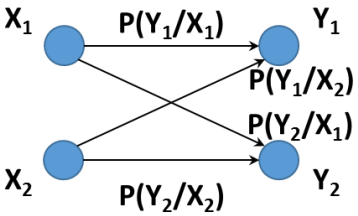
$H_{X/Y}$ - average lost information

$H_{Y/X}$ - average disturbing information

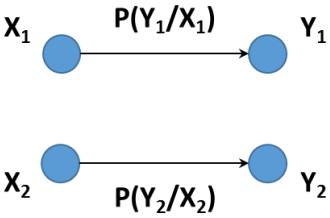
$I_{X \to Y}$ - mutual information

For channel information, we distinguish two types of channels:

- **Noisy:**



- **Noiseless:**



P (Y_1 / X_1) probability when sending element X_1 Y_1 received

P (Y_2 / X_2) probability when sending element X_2 Y_2 receive

P (Y_1 / X_2) probability when sending element $_2$ receive X Y_1

P (Y_2 / X_1) the probability that we get Y_2 when sending element X_1

Channel permeability (channel capacity) - Channel ability to transmit information. Maximum information that can be transferred per unit of time.

$C = B\left(1 + \frac{S}{N}\right)$, where B is the channel width, S is the signal, N is noise.

Sampling theorem

An accurate reconstruction of a continuous, frequency restricted signal from its samples is possible if the sampling rate is greater than twice the highest harmonic component of the sampled signal.

The minimum possible length of signal elements

$\tau_0 = \frac{1}{2F_m}$, F_m is the limit frequency, bandwidth

9. Information theory application

For example, Huffman encoding, arithmetic coding, LZW, JPEG, MP3, TIFF, error detection and error correction codes, statistical applications, and MDL are included in the theory of information.

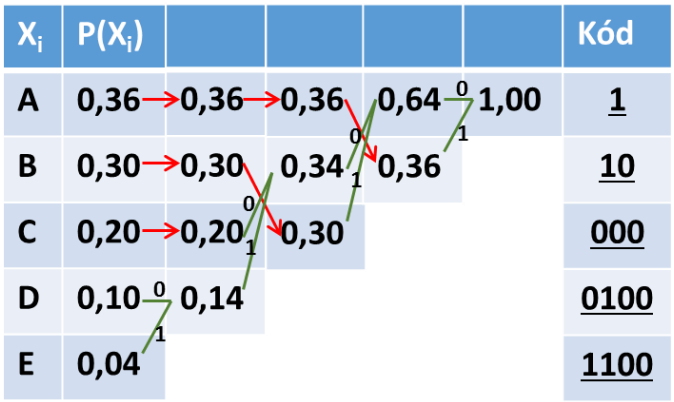
Huffman encoding is an algorithm for lossless data compression. Converts the characters input file into bit strings of different lengths. The most frequent symbols into bit strings with the shortest length (even 1bit). Least frequent to longer chains (can be longer than 8 bit).

It has two phases

- Passes the file and creates statistics
- Creates a binary tree to compress data

The code is made from the leaves to the root.

Example:



Shannon-Fano coding is the statistical method of lossless compression. Huffman's encoding differs only in the binary tree design. The character set is recursively divided into two roughly equal subsets. One sub-code is then assigned a binary 1 and the second 0. The code is thus constructed from the root to the leaves, unlike Huffman coding, the code is formed from the leaves to the root, may not be optimal.

Example:

Char	p(x)	s	Groups				Code
A	0,36	1	0				0
B	0,3	0,64	1	0			10
C	0,2	0,34		1	0		110
D	0,1	0,14			1	0	1110
E	0,04	0,04				1	1111

Arithmetic coding is a batch-length compression of variable length data of a code word. It encodes the entire text into a single number, a fraction $n \in < 0; 1)$. It is in the propeller difficult to count with real numbers. For a longer report, we would no longer be able to achieve the necessary precision, and some subintervals might begin to fuse us. It uses block compression, where the blocks are large enough to ensure sufficient accuracy when distributing.

Information can be secured during transmission to detect and correct errors and to prevent unauthorized reading.

Codes for error detection and correction

Parity - Data For each segment is connected to another bit, whose value of the number of binary ones complement of the number is odd or even (odd / even parity)

Even: 10011011 10010101 → 100110111 100101010

Odd: 10011011 10010101 → 100110110 100101011

CRC - checksum - The data is divided into sections of the required length (8, 16, 32 bits) and these segments are added after the bits without transfer. The resulting data segment connects to data transferred.

00001101	Data
00010000	
01000100	
10010000	
11110001	**CRC**

Hamming code is a linear code used in telecommunications to detect up to two erroneous bits or to repair one bad bit

Algorithm:

All bit positions whose number is equal to Power 2 are used for the parity bit (1, 2, 4, 8, 16, 32, ...).

All other bit positions belong to the encoded information word (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, ...).

Each parity bit is calculated from some of the information word bits. The position of the parity bit indicates the sequence of bits that are scanned in the code word and which are skipped.

For parity bit p_1 (position 1), in the remainder of the codeword, 1 bit is skipped, 1 checks, 1 skips, 1 checks,

For parity bit p_2 (position 2) the first bit is skipped, 2 checks, 2 skips, 2 checks, etc.

For p_3 (position 4), the first 3 bits are skipped, 4 are checked, 4 are skipped, 4 is checked,

Against unauthorized reading

- Encryption – Cryptography, Historical
- Stenography - Hiding the Message
- Substitution Cipher - replacing each character with another by some rule
- Character shift - Caesar's cipher - each letter of the alphabet shifted to a fixed number of positions
- Tables of substitutions - replacing a character with another without any internal context or knowledge of the key
- Vigener Cipher - uses passwords whose characters determine the offset of the open text, so that the open text is divided into blocks of characters long as the password and each character is added with the corresponding password character
- Verman's cipher - the only known cipher that has been proven to be undetectable so far is based on the addition of open-text letters and passwords, but the password is a block of randomly selected data of the same size as the open text

Transposition grid

- Skytalé - a type of encryption that consists of a war and a wound paper or parchment on it, on which a message is written Modern
- Symmetric ciphers - a conventional cipher, uses a single key to encrypt and decrypt, DES (Data Encryption Standard), the current AES (Advanced Encryption Standard)
- Asymmetric ciphers - Use different keys (private and public key), RSA, PGP (OpenPGP) for encryption and decryption , used for electronic signature
- Hash function - a one-way mathematical function that is used, for example, to ensure data integrity.

Signing

- Electronic Signature - O of the learning of specific data that replaces a classic handwritten signature or a certified signature on the computer. It is connected to or logically connected to a data message, it enables authentication of the signed person in relation to the data message. An electronic signature is a means of verifying the identity of the sender.

10. Theory of Complexity

The complexity theory focuses on classifying computational problems according to their own complexity and the relationship between classes. The problem is a task that can be resolved on a computer. The problem is considered to be difficult if its solution requires considerable resources no matter what algorithm is used. Formalizes this approach, introduces computational models to study and quantify the amount of resources needed to solve problems (time, memory). Another level of complexity is a lot of communication, gate count of the circuit, the number of accesses to the cache and the number of processors. One of the goals is to determine the practical limits of what computers can calculate and what they do not.

Analysis of algorithms vs. theory of complexity vs. the theory of computability

- Algorithm analysis deals with the amount of resources needed by a particular algorithm
- The complexity theory identifies more general questions about all the algorithms that can be used to solve a particular problem. Trying to classify problems according to the limitations of available resources, introduces restrictions on available resources
- The quantification theory asks what problems can, in principle, be solved algorithmically.

The basis is a computational problem.

Problem: Infinite collection of examples and solutions to the situation, entering input, ie. An instance of the problem cannot be confused with the problem itself. The problem must be addressed regardless of its assignment.

Example: Prime Number Test – Input: Number; Output: Yes / No

You can enter the problem in several ways: The most basic way is in the form of a pronounced sentence. For solving computer is necessary translation into the language of computers. Mathematical tasks in graphs are set to for example by means of adjacent matrices.

The decision problem is the basic type of problem complexity theory, has only two outputs Yes / No. In the language where the members of the language are the cases with the answer YES and the other members with the NO answer you should be using an algorithm to decide how the specified input corresponds with this language. If YES, then the algorithm returns the input is accepted otherwise rejected. The algorithm calculates a characteristic feature of the language.

Formal languages (and problems over them) are some of the alphabet, which is not necessarily binary.

The problem function is a computational problem where one output of the total function applies to all possible inputs but is more complex than the output of the decision function. The problem of the feature is richer than the decision-making problem. It can also be redesigned to the decision-making problem as follows: we want to multiply two

numbers, the answer to the decision problem is a triple (a, b, c) and the YES return is only for the triple where a * b = c

For complexity theory, you need to measure input size. The size of the input depends on the amount of time it takes to process the algorithm. These partial problems, which may be the space needed for the solution, process a separate algorithm and all relate to the size of bit entry. The complexity theory is concerned with how the algorithm will deal with the input size.

Ex.: solving connected graph of *n* edges in comparison with the graph of 2 *n* edges. If the input is *n,* then the time required to calculate the function (n) . If function $\tau(n)$ is polynomial we are talking about polynomial algorithm.

Cobham thesis - the problem can be solved in polynomial time, if it exists for the algorithm that processes the *n* -bit input in time, where *c* is a constant depending on the problem, not its input.

However, we measure not only inputs, but also the resources, whether p ro a specific algorithm or a problem.

Generally, the data size *n* , and not for a specific input value *to the* (size *n*), usually for all possible (infinite number) → size and complexity estimate usually asymptotically. IN resource dependency is measured by:

- Elapsed time (in steps)
- Memory (in bits / bytes / cell)
- Packets (generally in frames)
- Cache (e.g. the number of accesses)

Complexity Theory class defines the complexity of the problems:

- class P
  The decision-making problem of the U lies in the class *P* if there is a Turing machine that decides the language L_U in polynomial time.

  Ex .: Finding the shortest path, minimum spanning tree

- class NP
  The decision-making role of the *U* is in the class *NP* if and only if there is a non-deterministic Turing machine that decides the language L_U in polynomial time.

  Ex .: K-color (color chart can be given up *to* colors ?) Clique of graph (exists clique in graph at least of *k* vertices?)

- NPC class - NP-complete problem
  The problem is in the class *NP* and also true that the polynomial reduces the role of each class *NP* . *NP* -complete tasks are the "most difficult" of all *NP* problems.

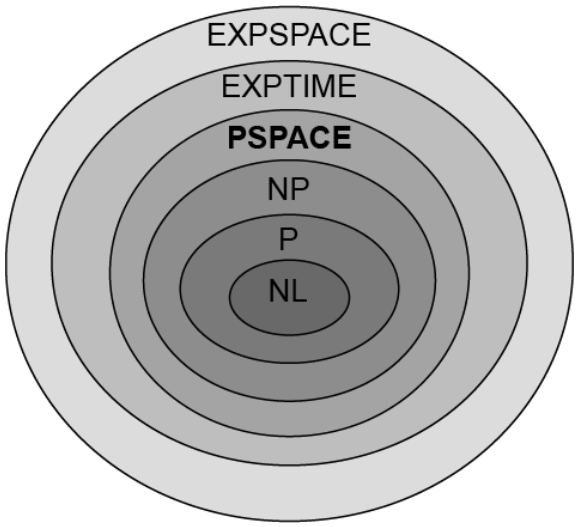
Ex .: Traveling Salesman Problem, Knapsack Problem

Classes PSPACE and NPSPACE

The language *L* is in the class *PSPACE* if and only if there is a deterministic Turing machine that works with the memory polynomial complexity (ie. Not use any memory cell index greater than *p (n)*) and adopts the language *L* .

The language *L* is in a class *NPSPACE* just when there is non-deterministic Turing machine that works with polynomial complexity of memory and accepts the language *L* .

It has been shown that $NP \subseteq NPSPACE$ a further according *Savitch sentences* , the $PSPACE = NPSPACE$.



11. Languages and automata

Languages and automata are the cornerstone of theoretical computer science.

The language means any nonempty set V as alphabet and its elements are characters or symbols.

We define:

- The word above the alphabet V is the final sequence of characters from V, w = a1 a2...an
- Word length w - length of sequence w, | w | = n
- Empty word ε - sequence of length 0
- A set of all words above the alphabet V ^ *
- A set of all non-empty words above the alphabet V ^ +
- Grammar G is quadruple (N, Σ, P, S),
    - N is the final set of nonterminal symbols (nonterminals).
    - Σ is the final set of terminal symbols so that no symbol belongs to N and Σ at the same time.
    - P is the final set of derivation rules. Each rule is the form (ΣUN) ^ * N (ΣUN) ^ * → (ΣUN) ^ *
    - S is an element of N called initial symbol.

Convention

- We denote terminals - a, b, c, ...
- Chains terminals denoted - u, v, w, ...
- Individual nonterminals - A, B, C ... X, Y, Z
- Strings of nonterminals and terminals - α, β, γ, ...
- An empty string is denoted by the symbol e or even ε

Grammar is divided into several types according to the Chomsky hierarchy.

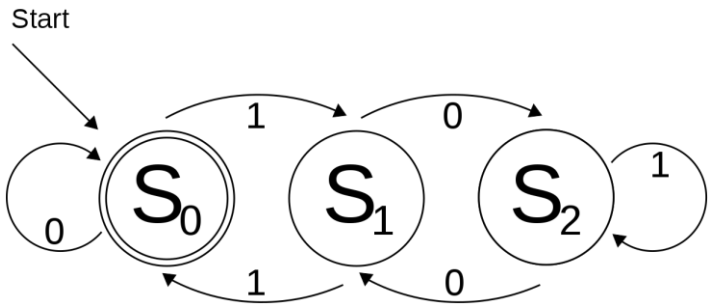
- Type 0 - All formal grammars (unrestricted grammars)
- Type 1 - Context grammars
- Type 2 - Context-free grammars
- Type 3 - Regular grammars

Finite automaton is the theoretical computational model used in computer science to study formal languages. It describes a very simple computer that can be in one of several states, between which it switches on symbols that read from the input. Set of states is finite (hence the name), finite automaton has no additional memory, in addition to the current status. We distinguish Deterministic FA - at each point in the table has just one target state - and nondeterministic FA - at every point of the table is not a target state, but the whole set of states. Also, in the transition table, there is an empty entry column, called e. Any non-deterministic machine can be converted to deterministic. The original set of states needs to be replaced by its potential set. Each state of the machine thus created corresponds to a set of states of the original non-deterministic machine and

there are clear transitions between them.

Formally, the finite automaton is defined as a ordered five (S, Σ, σ, s, A) where:

- S is a finite non-empty set of states.
- Σ is a finite non-empty set of input symbols, called alphabet.
- σ is the so called transition function (also a transition table) describing the rules of transitions between states. May either have $S \times \Sigma \rightarrow S$ (deterministic automaton) or $S \times \{\Sigma \cup \varepsilon\} \rightarrow P(S)$ (nondeterministic automaton), see below.
- s is the initial state, $s \in S$.
- A is a set of receiving states, $A \subseteq S$.



Activities of the machine:

Initially, the machine is in a defined initial state. At each step, reads a symbol from the input and enters a state which is given a value which, in the transition table corresponds to the current status and read back symbol. It continues reading the next symbol of the input, another transition by transition table etc. Depending on whether the machine stops after reading the input in a state that belongs to the set of recipient states, the automaton either the input accepted or not accepted. The set of all strings that the automaton accepts, form a regular language.

12. Turing machines

Church (Church-Turing) thesis: Turing machines (and their equivalent systems), define their computing power what intuitively consider to be efficiently computable. For each algorithm there is an equivalent Turing machine.

Church's (Church-Turing) argument cannot be formally proved, but is supported by a number of arguments:

- Turing machines are very robust - various modifications do not alter their computing power
- It suggested a number of different computational models whose strength corresponds to Turing machines
- There is no known computational process which we would describe as effectively quantify and which would not be possible to implement the Turing machine

Turing machine is the theoretical model of a computer. It consists of several parts:

- Processor unit - finite automaton
- Program - Transition function rules
- Right-side endless tape - Used to record intermediate results

It is used for modeling algorithms in the theory of computability.

Turing's complete are the programming languages and computers that have the same computational power as Turing's machine

Definition:

$M = (Q, \Gamma, b, \Sigma, s, \delta, F)$	Turing machine
Q	Finite set of internal states
Γ	Finite alphabet of symbols on the tape
$b \in \Gamma$	An empty symbol is not part of the input string's alphabet
$\Sigma \subseteq \Gamma \setminus b$	Finite set of input symbols
$s \in Q$	Initial condition
$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$	Transition function, L Move left to left, R Move the head to the right
$F \subseteq Q$	Set of final states
Configuration:	$\langle q, s, n \rangle \in Q \times \{yb^{\omega} \mid y \in \Gamma^{*}\} \times N_0$
q	Current status
s	The smallest continuous part of the tape containing the non-empty symbols
n	Read head position (cell number)

If Q, Γ are disjoint sets, a compact shape can be used

Tape: 1234

Condition of the machine: q

Reader Head Position: 2

Configuration: 1q234

Initial configuration of TM input $w \in \Gamma^*$

$(s, wb^\omega, 0)$, Compact form: sw

Calculation method for Turing machine:

If the current status is the end state, it finishes the calculation

The read head reads one symbol from the cell on which it is currently located

If there is a transition defined in the transition function for the current state and a transition is defined for the symbol being read, then (in the case of multiple possible transitions in non-deterministic machines, one is chosen randomly):

It will change the status

The appropriate symbol is entered on the current head position

The head moves accordingly (does not shift)

There are various modifications of TM.

TM with the possibility to perform the calculation without head shifting

Transitional function extended by $N - \delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$

The N shift (no sweep of the read head) can be arranged on a conventional TS so that the read head reads the input symbol from the cell on which it is located, performs the appropriate operation and moves to the right (R). Now reads the symbol from the cell, writes the same symbol (i.e. does not change the symbol on which the head is located) and moves left (L). That's how we got the read head to the previous location and no other symbol was changed.

TM with two-sided endless tape

The tape is not left bounded

Possible move of the read head to the left of any configuration

N-tape TM

Reads from and writes to multiple tapes at once

The only change is in the transition function: $\delta: Q \times \Gamma^n \to Q \times (\Gamma \times \{L, R, N\})^n$

Non-deterministic TM (NTM)

Allows "multiple choice"

$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R,N\}}$$

Subroutines are used to facilitate the creation of complex TS. The subroutine is the set of states that contains the initial and final state. Usually it solves a partial problem in TS. In normal programming, it is the equivalent of a function.

The universal TS - U as input accepts the code of another TS T and the machine input word T. It decodes the transition function of the machine T and simulates the calculation of the machine. It can compute any partial recursive function (or is equivalent to universal partial recursive function), decides any recursive language and accepts any recursively recurring language.

13. Computability theory

Basic theory of computability question is: What is and what is not algorithmically computable (solve)? There are problems for which algorithms to solve them?

The problem is defined by:

- Name: XY
- Instance: what can be input?
- Result: output, input to output relationship

The problem is partial display type$\Sigma^* \rightarrow \Sigma^*$, valid for entry gives a valid output for invalid input special outcome "Incorrect input".

Problems have the following characteristics:

Algorithmic solvability

For a given problem, we say that it is algorithmically solvable (or the respective (partial) view $\Sigma^* \rightarrow \Sigma^*$ is algorithmically computable) if there is an algorithm which is capable as an input to accept any instance of the problem and its calculation for any such entry ever ends, the outputs of the desired result.

Algorithmic decidability

About the problem of Yes / No we say that the algorithm may decidable if there is an algorithm that is able to accept as input any instance of the problem and its calculation for any such entry ever ends, the output will be required answer Yes / No.

The problem is algorithmically solvable if there is an algorithm that for each particular task is always a problem in a specific time clearly determines the desired result. Decidability problems for a Yes / No

Partial decidability

About the problem of Yes / No we will say that is partially decidable if there is an algorithm whose calculation is over just for those entries that match the instances of the problem with the answer Yes.

The problem is partially decidable, if you know the algorithm stops and gives a result only if the answer to the question is yes. If the answer is No, so she'll never know, because the algorithm never stops.

Complexity theory provides us with interesting results. One of them is called Halting problem. We cannot construct an algorithm that would verify the finality of the general program of its run.

Complexity theory provides us with interesting hypothesis. One of them is the Church-Turing thesis. For each algorithm, there is an equivalent Turing machine.

Halting problem

Assignment: If you know the source code of the program and its input, decide whether the program stops, or whether to run forever without stopping.

In 1936 Alan Turing proved that there is no general algorithm that would solve the halting problem for all the inputs of all the programs there. Halting problem is therefore referred to as algorithmically undecidable problem.

Proof:

Undecidability of halting problem can be proved by contradiction.

Initial assumption: Assume that the halting problem can be decided. This means that there is a program stops (program, input), which is a universal solution to the problem of stopping - we assume, therefore, that if this program we will pass any program and its input, the program stops in a finite time returns the response such that if the calling program (input) after a finite number of steps over, then stops (program, input) returns YES, otherwise (if in Managing Calls program (input) trapped in a loop) returns Sun.

Subsequently Construct program Paradox (program) that calls Stops (software program), and if the call returned YES, deliberately zacyk divided, and if returned NE and ends immediately.

We ask, what is the result of a call Paradox (Paradox).

Assume for a moment that stops (Paradox, Paradox) returns YES. By definition, program Paradox then we know that Paradox (Paradox) loop forever. By definition, the program stops but, if the Paradox (Paradox) loop forever, then it must stop (Paradox, Paradox) returning Sun. We have come to dispute with.

On the other hand, suppose for a moment that stops (Paradox, Paradox) returns Sun. By definition, program Paradox then we know that Paradox (Paradox) stops. By definition, the program stops but, if the Paradox (Paradox) stops, then it must stop (Paradox, Paradox) return YES. Again, we came to the dispute.

Since we exhausted all possibilities and always reached the dispute shall be false initial premise. Consequently, the program stops (program, input), as defined, does not exist.

Church - Turing thesis

The hypothesis says that any possible calculation can be successfully carried out an algorithm running on the machine if there is sufficient time and memory.

The algorithm must meet the following requirements:

The algorithm consists of a finite number of instructions, which are precisely defined by using a finite number of symbols.

The algorithm always returns the result after a finite number of steps.

The algorithm can perform even a man with a pencil and paper.

Running the algorithm does not need human intelligence, except that which is necessary to understand and execute instructions.

Since every computer program can translate it into the language of a Turing machine, and vice versa, can be a thesis equivalent to formulate any commonly used programming language.

The programming language requires one of the following construct s (among others) that a Turing-complete (i.e. Equivalent Turing machine)

Cycle while -do.

Unlimited (at least in theory) recursion,

Conditional jump.

Common programming languages tend to have all three of these structures. Among the languages that are Turing- complete not include SQL (meaning no stored procedures.

ALGORITHMS AND DATA STRUCTURES

1. Algorithm

By the term algorithm is meant a precise instruction or procedure that can solve the given job type. It describes the theoretical principle of the solution, unlike the exact (specific) entry in the given programming language.

The algorithm should have certain properties:

Finality - Each algorithm must end in the final number of steps. There can be any number of steps, but it must be final for each individual input. A procedure that does not meet this condition cannot be called an algorithm, but only a computational method.

Generality - The algorithm does not address one particular problem, but a general class of similar problems.

Determination - Each step of the algorithm must be unambiguously and precisely defined. In each situation, it must be clear what to do and how to do it, how the algorithm should continue. Some algorithms are not determined - they contain a random element (eg: genetic algorithms)

Output (or Resultativity) - The algorithm has at least one output, the quantity that is in the required relation to the inputs. The output is the answer to the problem.

Elementarity - the algorithm consists of a finite number of simple (elementary) steps.

1.1. Algorithm development process

There are several procedures that are used to design algorithms.

Top-down method - process solutions are decomposed into simpler operations until it reaches the elementary steps.

Bottom-up method - from elementary steps, we create resources that will eventually make it possible to handle the specific problem.

Alternatively, a **combination of the two methods**, when the top-down method is completed by "a partial bottom-up step" (usin the function library, high-level programming language or system programming ...).

Usual procedures for algorithms are the following methods: Divide and conquer, greedy algorithms, dynamic programming and backtracking.

- **The Divide and conquer method** divides the problem into sub-tasks that must be independent of each other. Then these sub-tasks are solved. It is often implemented recursively or iteratively.

- **Greedy algorithm** is mostly used for solving optimization problems. It always selects a local minimum (maximum) in an attempt to find a global minimum (maximum) . This process is not always ideal and may not reach the global minimum (maximum)

Dynamic programming divides the problem into subtasks, just like the Divide and conquer method, but in such a case these parts may be dependent.

Backtracking, or search return, is a way to solve algorithmic problems based on searching the status tree of the problem. It is a brute-force search.

1.2. Types of algorithms

Algorithms can be divided into several types.

- **Recursive algorithms** that use (call) themselves.
- **Randomized algorithms** that makes some decisions of random (pseudo-random) choices.
- **Parallel algorithms** that allocate tasks between more computers (processors, threads).
- Another type is **genetic algorithms** based on the imitation of biological evolutionary processes.
- **Heuristic algorithms** that do not search for a precise specific solution, but just some appropriate approximation. They are used in such situations where available resources are insufficient to be used for exact algorithms or no suitable exact algorithms are known at all.

2. Abstract Data Type – ADT

2.1. Description

Abstract Data Type - ADT - is an expression used for data types that are independent of their own implementation.

Using ADT, we try to simplify and streamline the program that performs operations with the given data type.

Each ADT can be deployed using basic algorithmic operations such as assignment, addition, multiplication, conditional jump,

2.2. Properties of ADT

ADT should have the following properties:

- **Generality of implementation -** Once designed, ADT can be embedded and run smoothly in any program.
- **Exact description** - The link between the implementation and the interface must be definite and complete.
- **Simplicity** - The user should not be involved in internal implementation and administration of ADT in memory.
- **Encapsulation** - The interface as a closed part, the user knows what ADT does, but not how it works
- **Integrity** - The user cannot intervene in the internal data structure
- **Modularity** - The "modular" programming principle is well-arranged and allows easy code parts exchange. When searching for errors, the individual modules can be considered compact units. There is no need to intervene in the whole program to improve ADT.

If ADT is programmed object-oriented, then these properties are typically met by default.

2.3. Operation in ADT

Using ADT, basic operations can be performed. These include the Constructor, Selector, and Modifier:

- The **constructor** creates a new ADT, assembles a valid internal representation of the value based on the given parameters.
- The **selector** obtains the values that make up the components or properties of a specific ADT value.
- A **modifier** performs changes to data type values.

3. Algorithms analysis

3.1. Process efficiency comparison

Since one problem can usually be solved using several different procedures (algorithms), it is necessary to have a tool that allows us to compare the effectiveness of the process. Algorithms can be compared either experimentally or theoretically.

Experimental analysis is time consuming. Of course, time required increases with the amount and size of inputs. This type of analysis requires a specific implementation of the algorithm, which obviously requires additional knowledge. It's hard to find an average case. Therefore, it is better to concentrate on the worst case possible that is easy to analyze and is critical for most applications, whether it's games, finance, robotics, and automated operations.

Experimental analysis of time required takes place in the environment where the algorithm (program) is being implemented mostly using an internal time measurement function. The program run depends on the inputs and their composition, and not all inputs are included in each program run. Comparing two algorithms requires the same hardware and software and the same memory occupancy.

Instead of experimental analysis, certain theoretical procedures can be used. The theoretical analysis uses the description of the algorithm by means of operations instead of a specific implementation. It takes into account all inputs and enables to rate the algorithm speed independently of hardware or software.

One of these tools is the so-called Pseudocode. Pseudocode allows to use higher levels of algorithm description. The description is more structured than a commonly written text, but less detailed than a specific implementation. It is a preferred type of writing for describing algorithms. Its advantage and disadvantage is that it hides problems of a particular implementation.

The pseudocode uses keywords to describe the algorithm:

For run control: -If...then...else (condition), while...do, repeat...until, for...do (cycles)

Header: Algorithmus Name(arg1, arg2...), input, output

Procedure call (Methoden, Algorithmus): var.Name(arg1,arg2,...)

Value return: return *Expression*

Expressions:	←	Zuschreibung
	=	Gleichheit
	+, -, n^2,...	Mathematische Operationen

3.2. Primitive operations

Primitive operation is a basic operation performed by algorithm, identifiable in a pseudo-code, independent of the programming language, and should be precisely defined. Such a primitive operation may nr expression evaluation, assigning a value to a variable, index it in a field, procedure call or returned.

Similarly, we can use asymptotic notation (big O, Bachmann-Landau notation). It determines the operational demandingness of the algorithm by determining how the algorithm behavior will change depending on the size of the input data. Asymptotic time and space complexity is usually used. The type of the write used means that the algorithm demandingness is less than $A+B \cdot f(N)$, where A and B are appropriately selected constants, and N is the variable describing the size of the input data. The multiplicative and additive constants, ie. $O(N+1000)=O(1000 \cdot N)=O(N)$ are disregarded. We are only interested in the behavior of a large N values.

To determine the time required for the algorithm using the big O notation, we must find the largest possible number of primitive operations, which we then express by means of the big O notation.

4. Queues and stacks

4.1. Stack

A stack is a data structure that serves to store data. It is characterized by a way of data manipulation - it accesses the data using the LIFO (Last In First Out) principle. It can be imagined as a plate container.

The ADT stack must contain at least:

- operations for inserting an object,
- returning and removing the last object,
- querying on the top of the stack,
- its size,
- whether the stack is empty.

If we try to perform a pop or top operation on an empty stack, we will get an Empty-StackException exception.

Stack application is, for example, the history of the web browser, the Undo sequence in the editors, or the individual procedures call. The stack can be used as an auxiliary data structure for other algorithms or as part of other data structures.

The simplest way to implement the sis by means of an array. We add the elements from left to right and the auxiliary variable holds the index of the last element.

Thanks to the array properties we get the following properties:

- *n - number of elements in the stack*
- Memory requirements - *O (n)*
- Time requirement of each operation - *O (1)*

Array also brings some limitations:

- At the beginning, we must define the stack size
- stack size cannot be changed at will
- Adding an element to the full stack will cause an implementation-specific exception

4.2. Fronta

The queue data structure FIFO (First In First Out). The minimum queue implementation must include operations for:

- inserting an item at the end of the queue,
- selecting an item from the beginning of the queue,
- query on the beginning of the queue,
- its length and occupancy.

Llike the stack, the queue can throw an exception during the dequeue operation or queues over a blank stack?queue - EmptyStackException.

Queue application is for example waitlist, queue, access to shared resources (printers), multiprogramming. The queue can also be used as an auxiliary data structure for other algorithms or as a part of other data structures.

The queue can be implemented using an array. To improve its properties, a circular array is used. We thus have two variables, f - index of the first element, r - index of the last element plus one (pointing to the first free space).

5. Vectors, Lists, Sequences

5.1. Vectors

The vector extends the concept of array by storing the sequence of any objects. The element in the vector can be read, inserted and removed by determining its order The vector enables basic operations:

- elements in a specific order,
- replacing the element at a specific location, inserting it into a particular location,
- removing an element from a particular location,
- enables to determine the size and whether the vector is empty

Vectors operations may throw an exception in case of an incorrect index (usually negative). Vector application is a sorted object collection (basic database). Vectors can be implemented using an array. This brings the following properties:

- *The variable n indicates the length of the vector*
- *Operation isEmpty () elemAtRank (r) replaceAtRank (r, O) - time complexity O(1)*
- *Operation insertAtRank (r, O) - Time complexity O (n)*
- *Operation removeAtRank (R) - Time complexity O (n)*

5.2. Lists

Another data structure is the list. The list is a sequence of positions storing any data. It introduces relationships before / after between positions.

General operations are a query on size, and a blank list query. Other operations are finding out whether the given element is the first or the last one, obtaining the first and the last element, the preceding or the following element.

The ADT list contains:

- replaceElement(p, o),
- swapElements(p, q),
- insertBefore(p, o),
- insertAfter(p, o),
- insertFirst(o),
- insertLast(o),
- remove(p).

Lists can be divided into single linked list and double linked list. In a single linked list, the element contains a reference to the next node; in the case of a double linked list, the element also contains a reference to the preceding node.

5.3. Sequence

ADT sequence is a combination of a vector and a list, the elements can thus be accessed using both the position and sequence. Besides vector and list operations, it also includes interconnecting operations **atRank (r)** and **rankOf (p).**

Sequence is a general basic type that can be used to store an arranged set of elements. It is a general replacement for a stack, queue, vector or list. It can also be used as a small database.

6. Trees

Trees represent a model of a hierarchical structure that consists of nodes with a parent/child relationship between them.

Trees can be used as an organizational chart, file systems, or programming environment.

The following terminology is used to describe trees and their parts:

6.1. Types of nodes

- Root (root)
- Inner node - a node that is neither a root nor leaf
- List (leaf node, external node) - a node that has no offsprings
- Parent node - the node directly preceding the node on the path from leaf to root
- Child node - the node that follows directly after the node on the path from the root to the leaf
- Sibling - siblings refer to as the nodes with the same parent
- Ancestor node, predecessor node - a node that lies in front ofthe given node on the path to the root (the nearest ancestor is the parent)
- Successor node - a node that lies behind the given node on the path from the root to any leaf (the next descendant is a descendant)
- Depth - tree depth is the length of the longest path from root to sheet, with an empty tree defined as -1
- Level - is usually used for a set of nodes located at the same distance from the root, counted by the number of nodes

6.2. Structure

- Subtree – is a subgraph of a tree, which is also a tree (most frequently, there are subtrees formed by taking a tree node as a new root and retaining the rest of the structure,
- Branch - Each path from root to leaf.

6.3. Operations for Trees manipulation

General operations:

- integer size(),
- boolean isEmpty(),
- objectIterator elements(),
- positionIterator position().

Access operations:

- position root(),
- position parent(),
- positionIterator children(p),
- Dotazovací operace,
- boolean isInternal(p),
- boolean isExternal(p),
- boolean isRoot(p).

Actualization operations

- swapElements(p, q),
- object replaceElement(p, o).

Since tree is a hierarchical structure, it can be tracked in several ways.

Pre-order traversal

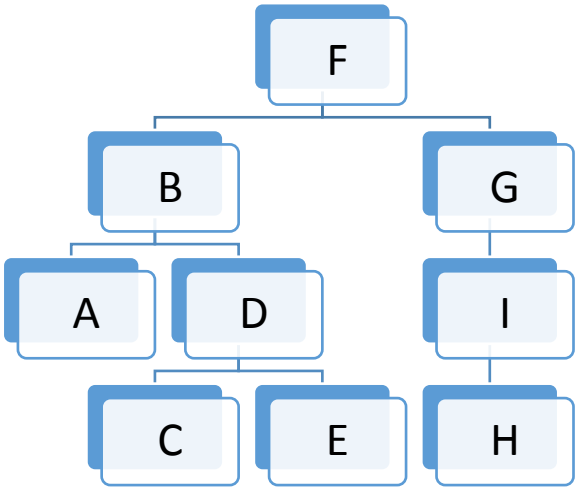
- Checking whether the node is empty or null
- Displaying the current node data
- The left subtree throughput by recursive pre-order function call
- The right subtree throughput by recursive pre-order functions call

In-order throughput

- Checking whether the node is empty or null
- The left subtree throughput by recursive pre-order function call
- Displaying the current node data The right subtree throughput by recursive pre-order functions call

Post-order throughput

- Checking whether the node is empty or null
- The left subtree throughput by recursive pre-order function call
- The right subtree throughput by recursive pre-order functions call
- Displaying the current node data



Each type of throughput gives different results.

- Pre-order: F, B, A, D, C, E, G, I, H
- In-order: A, B, C, D, E, F, G, H, I
- Post-order: A, C, E, D, B, H, I, G, F

Using the ADT Tree, it is also possibleto define a Binary Tree or other types.

The binary tree extends the definition of the tree by the fact that each node has no more than two children, forming an ordered pair (left descendant, true descendant).

The binary tree adds additional operations:

- position leftChild(p)
- position rightChild(p)
- position sibling(p)

7. Priority Queue and Heap

7.1. Priority queue

Priority queue stores a collection of items where the item is ordered pair key (priority) -the value.

Basic implementation should include:

- **insertItem(k, o),**
- **removeMin(),**
- **minKey(k, o),**
- **minElement(),**
- **size(),**
- **isEmpty().**

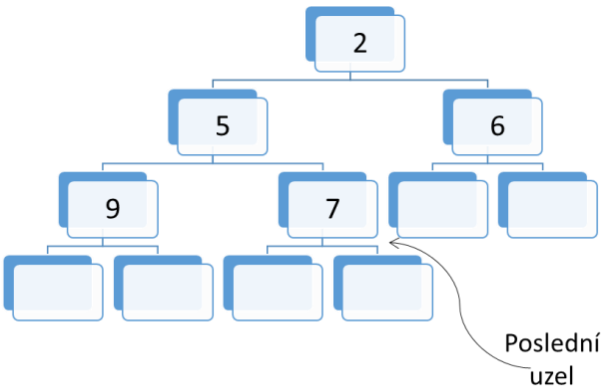
The priority queue application is for example auctions and stock exchanges. The priority queue key can be any object with a defined order and which can be arranged. Two different elements (values) can have the same key (priority).

To use the priority queue, the ADT Comparator must be implemented, which enables to compare two objects.

7.2. Heap

Heap is a binary tree that stores keys as internal nodes. For each tree node outside the root it holds true that the node key is larger than the parent key. For each heap a complete binary tree is defined. If *h* is the height of the tree, then for *i* from 0 to *h-1* there is 2^i nodes of *i* depth.

The last heap node is an internal node, which is located at the rightmost at the *h -1* level.



Heap can be used to implement the priority queue. Then we store an item (key, value) in each internal node, and keep a reference to the position of the last element.

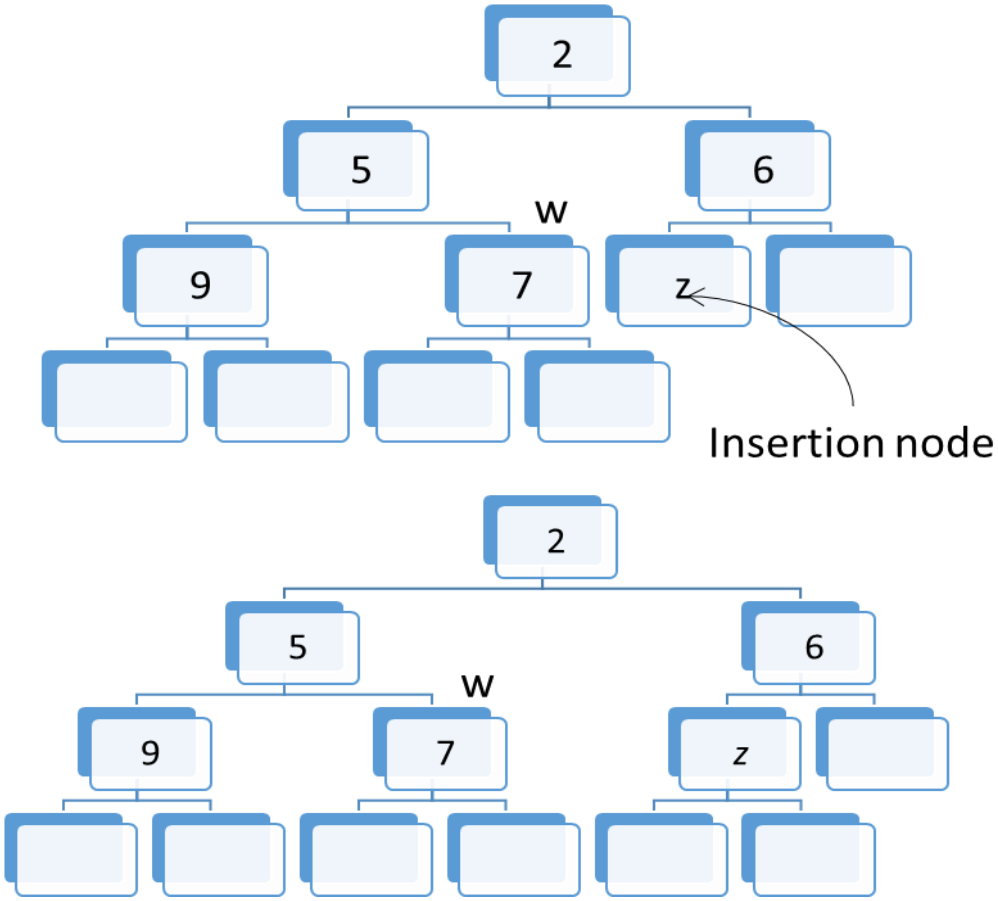
7.3. Handling a heap

Operace insertItem (k,o)

- Finding the node where it will be inserted
- Saving the k key into the z node, changing z node into internal node
- Recovering the heap order (checking properties) - upheap () operation

Operace removeMin ()

- It corresponds to removing the root from the heap (node 2)
- Changing the root for the last node (2-7)
- Changing the w node and its children into the list



upheap()

- inserting a node may affect the arrangement/layout
- Upheap algorithm restores the arrangement by swapping *k* key upwards from the inserted node
- It ends when the internal node becomes the root or parent key is less than or equal *k*

downheap()

- removing a node may affect the arrangement/layout
- downheap algorithm restores the arrangement by swapping k key downwards from the node
- it ends when the internal node becomes a leaf or the child key is greater than or equal *k*

8. Dictionaries and Hash Tables

8.1. Dictionary

Dictionary refers to the ADT containing a key-value collection that can be searched for. Operations that can be performed with the dictionary:

- **findElement(k),**
- **insertItem(k, o),**
- **removeElement(k),**
- **size(),**
- **isEmpty(),**
- **keys(),**
- **elements()**

Dictionary application is a directory, credit card authorization, dictionary, domain name translation to IP address.

Log File - dictionary implemented as a random sequence (double linked list). We store the objects in any order.

Complexity of operations:

- Vložení objektu O(1)
- Nalezení prvku, odebrání prvku O(n)

The log file is suitable for small dictionaries or applications where the most frequent operation is inserting, while searching and removing is rarely done.

The **findElement (k**) operation on a dictionary implemented by a sequence based on a key-based array is performed as a binary search. At each step, the candidate number is divided by two, ending after the logarithmic number of steps.

Lookup table is a Dictionary implemented using an arranged sequence. We store the dictionary entries in a sequence based on a key-based array; an external key comparator is required.

Complexity of operations:

- Finding the element O(log(n))
- Inserting element, removing element O(n)

This is effective for small dictionaries or applications where searching is a frequent operation.

Binary search tree is a binary tree for which the following rules apply:

- u, v and w are such nodes for which it holds true that u is in the left v subtree and w is in the right v subtree

- $key(u) \leq key(v) \leq key(w)$
- External nodes do not store any items.
- The in-order throughput gives the keys in the ascending order

8.2. Hash table

Hash function h is a function that allocates an integral value from the 0 and N-1 interval to the key of a given type between 0 and N-1. The purpose of this function is to split the keys uniformly at a given interval.

A hash table for a given key type contains a hash function and a N-size array (table).

The key is replaced by a hash value. However, it may happen that for the two keys the same hash value is generated - a collision occurs. This can be solved in two ways:

- by chaining - conflicting items are stored as sequences
- by open addressing - the item is saved elsewhere in the table.

9. Sorting algorithms

9.1. Explanation

Sorting algorithms are used to sort the data file in a specific order, either alphabetical or numerical. The key-value pair is ranked by the key and the value is not taken into account.

Sorting algorithms can be divided into stable and unstable, depending on whether they keep the order of items with the same key, natural and unnatural - natural work faster with a partially arranged set.

They can also be broken down by the sorting type:

- By selecting
- By inserting
- By replacing
- By merging

The best-known algorithms:

- Bubble sort
- Heap sort
- Insertion sort
- Merge sort
- Quicksort
- Selection sort

Other algorithms based on a different principle

- Bucket sort
- Radix sort
- Counting sort

9.2. Bubble sort

- Easy to implement.
- Universal, local (in-place, no need of extra memory).
- The algorithm starts at the beginning of the data set. It compares the first two elements, and if the first is greater than the second, it swaps them. It continues doing this for each pair of adjacent elements to the end of the data set. It then starts again with the first two elements, repeating until no swaps have occurred on the last pass.

Algorithm:

```
procedure bubbleSort( A : list sortable items )
    n = length(A)
    repeat
        swapped = false
        for i from 1 to n-1 inclusive do
     if A[i-1] > A[i] then
                        swap( A[i-1], A[i] )
                swapped = true
            end if
        end for
    until not swapped
end procedure
```

9.3. Heap sort

- A comparison-based sorting algorithm.
- Not stable.
- Uses the heap data structure and its properties.

Algorithm:

```
    procedure heapsort(a, count) is
        input: an unordered array a of length count
        heapify(a, count)
       end ← count - 1
        while end > 0 do
             swap(a[end], a[0])
            (the heap size is reduced by one)
            end ← end - 1
            (the swap ruined the heap property, so restore it)
            shiftDown(a, 0, end)
```

If the smallest element is the root – placed on the first place in the array and the root is removed.

Downheap() – recovering the heat following the rulesWe repeat removing the root and heap recovery until the heap is empty.

9.4. Insertion sort

- A simple sorting algorithm that builds the final sorted array (or list) one item at a time
- Simple implementation
- Efficient for (quite) small data sets
- Efficient for data sets that are already partly sorted
- Stable, on-line, in-place

Algorithm:

```
for i = 1 to length(A)
    j ← i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j ← j - 1
    end while
end for
```

9.5. Merge sort

- is an efficient, general-purpose, comparison-based sorting algorithm
- the divide and conquer method
- Stable, divide and conquer algorithm, easy to paralelized
- Worst and average time: O(NlogN)
- Extra memory needs: array of N size

Algoritmus

```
mergesort(m)
      var list left, right
      if length(m) ≤ 1
            return m
      else
           middle = length(m) / 2
           for each x in m up to middle
                add x to left
           for each x in m after middle
                add x to right
left = mergesort(left)
           right = mergesort(right)
           result = merge(left, right)
      return result
```

9.6. Quicksort

- an efficient sorting algorithm
- It takes: O(NlogN) – O(N2)
- Divide and conquer algorithm, not stable, in-place
- Recursive
- The steps are:
    - Pick a pivot from the array.
    - Partitioning: reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the partition operation.
    - Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.
- Pivot selection –median is ideal
    - First element (any fix position) – not suitable for partly arranged sets
    - Random element – actually pseudo-random
    - Median of three (five...) – or other number of elements from the fix or random positions

When pivot is selected properly, it does not need extra memory. Quicksort is an unstable algorithm. The pivot selection method affects sorting, but on average, it is the fastest universal array sort algorithm in the computer operational memory.

9.7. Selection sort

A simple algorithm whose time complexity is O (N 2),It is suitable for small data volumes. It is universal, local and unstable.

Procedure:

1. Dividing the sequence into an arranged and unarranged unassigned part.
2. Finding the element with the smallest value in the unarranged part of the sequence.
3. Replacing it with the element in the first position of the unarranged part.
4. The first element of the unarranged part is included in the arranged part and at the same time the unarranged part is reduced by 1 element from the left.
5. The remainder of the sequence is arranged by repeating steps 2 - 5 for the remaining unarranged part.
6.

Comparison of sorting algorithms:

Name	Time complexity			Extra memory	Stable	Natural	Method
	Minimum	Average	Maximum				
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	yes	yes	Exchange
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$	no	no	Heap, exchange
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	yes	yes	Inserting
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(\log n)$	yes	yes	Merging
Quicksort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$	no	no	Exchanging
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	no	no	Selection

9.8. Bucket sort

It divides data into several buckets, its time requirement is O (n * k), where k = n / m, input n, number of bucket is m.

To use Bucket sort, prerequisites are required:

- Suitable for evenly distributed input data values.
- The algorithm for arranging the buckets must be stable

Procedure:

- Input data are divided into a predefined number of buckets.
- For each bucket a stable sorting algorithm is called.
- The individual buckets are gradually copied into the output array.

The benefits of the bucket sort are that it is well parallelizable and does not require having all the data in memory at the same time.

9.9. Radix sort

It sorts integers by scanning through all digits. There are two approaches:

- LSD (Least Significant Digit)
- MSD (Most Significant Digit)

Time required: O((z+n)*$\log_z u$), where z is the basis of the selected number system, *n* are the numbers at the input and *u* is the maximum range of numbers at the input

It is not suitable for unlimited input size.

Example of LSD radix: 170, 45, 75, 90, 802, 2, 24, 66 ⇒ 170, 90, 802, 2, 24, 45, 75, 66 ⇒ 802, 2, 24, 45, 66, 170, 75, 90 ⇒2, 24, 45, 66, 75, 90, 170, 802

9.10. Counting sort

Suitable for large files with a small amount of discrete values; stable. The time required is *O(N+M)* and extra memory needs *O(M.)*

Prerequisites:

- Number of different values *(M)* is significantly smaller than the total number of elements *(N).*
- Auxiliary array - writing and reading at a constant time (array indexed by value or hash values).

Algorithm:

- Passing the input array from the left (or from the right).
- For each element, the frequency of occurrence of this element is increased in the auxiliary array.
- To each item, the number of occurrences of all previous items (it gets the exact position of the border) is added.
- Starting to pass the unarranged array from the right.
- For each element, it looks into the auxiliary array at the top of the placement boundary.
- Placing the element into that border and reduces it by one.
- Repeating until the entire array is passed.

10. Pattern matching

Pattern matching, is searching for a specific pattern in a given sequence, usually searching for a substring in a string

First, a string must be defined. String is a sequence of characters from the given alphabet. Alphabet is a set of all possible characters - Ascii, Unicode, {0,1}, {A, C, G, T}

If the *P* string length is *m,* then the substring *P [i..j]* of *P* string consists of characters between *i* and *j.* The string in front of the index *i* is prefix. The string located behind the index *j* is a suffix.

Application - text editors, search tools, biological research.

There are several algorithms for pattern matching.

The basic one is the **Brute-Force** (brute force).

It passes the text from left to right

It compares the *P* pattern with the T text *,* for every possible position until:

a match is found

all possible positions have been tested

Time required for this algorithm is *O (nm).*

Boyer-Moore algorithm goes through text from the end (right to left).

We define:

- The index i indicates the position in text T, the index j points to the P

During the search, there are 4 possible situations:

- *T (i)* is not in *P;* then i will be moved bythe *P* length *(P* aligned to the next letter *T,* that is, *T (i + 1))*
- *T (i)* corresponds to *P (j)* – we move in both to the left and repeat (as in the brute force)
- *T (i)* is *P (j) T (i)* is in front of the index *j P -> P* aligned to the right so that *T (i)* corresponds to the occurrence of *P* and repeat
- *T (i)* is *P (j) T (i) P* is an index *j* -> move to the right by 1 and repeat (not recurring, but not move more below)

Return *i* - if the whole pattern is found.

The algorithm is faster than the Brute-Force, however, its complexity can be *O (mn + A),* where A is the size of the alphabet.

Preprocessing is required to apply this algorithm. It finds out the position of the letters (from the left)

- If the patern is eg: "ABRAKADABRA"
- A gets index 10, B gets index 8, K = 4, D = 6 and R = 9. For other letters, we assign -1.
- We implement the function Last (char input) which returns the index according to the letter, then, for cases 3 and 4, we compare the last *(T (i))* and *j,* and thus we know whether to move it tothe Last *(T (i))* (if the Last is *(T (i)) <j)* or only *i ++*

Knuth-Morris-Pratt (KMP) algorithm

It searches thetext from left to right, unlike brute-force it does not make all comparisons. If a disagreement is found, it moves by more than one letter. If we find a P (from the beginning, a prefix), the characters of this prefix match the text, so no need to check them again. The end of the found substring can also be included at the beginning of this substring. Such a match is, of course, the whole of the P found, so we are looking for P + 1. So we go from the end of the found P piece from the left and right, and when we can not find a match, we know how much we can move. This can be computed into a table - then all is O (1).

Trie

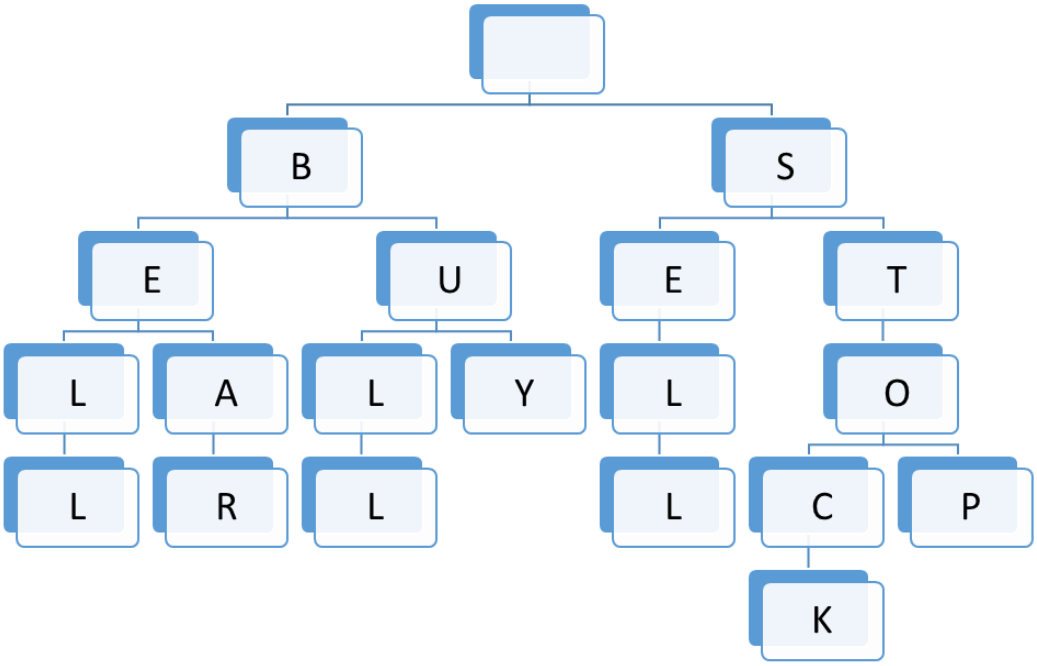
Trie is a tree structure for text pre-processing. Each node has one letter. The length of the path from the node to the top determines the position of the letter in the word.

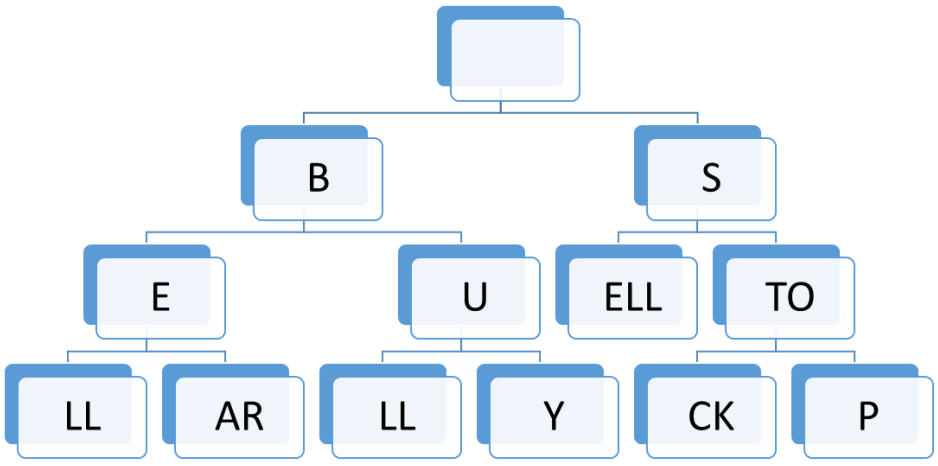
The search has a time requirement O (dm), where d is the alphabet size, m is the length of the word.

It is possibleto save all text in the Trie structure, then each node contains one word.

For saving,the so-called compressed Trie can be defined. The tree then contains nodes of at least degree two (two letters per node).

Example:S={BELL, BEAR, BULL, BUY, SELL, STOCK, STOP}


B
S
E
U
E
T
L
A
L
Y
L
O
L
R
L
L
C
P
K


B
S
E
U
ELL
TO
LL
AR
LL
Y
CK
P

11. Graph theory

Graph is an ordered pair (V, E) where V is the set of vertices and E is the set of edges. Each edge is determined by just two vertices, optionally by the direction or weight.

11.1. Types of edges

There are several types of edges:

- Oriented - ordered pair of vertices *(u, v)* where *u* is the beginning, *v* is the end/ target
- Unoriented - ordered pair of vertices *(u, v)*
- Loops - the edge begins and ends at the same vertex
- Multiple edge (multiple, parallel) there are more edgesbetween vertices *(u, v)*

11.2. Types of graphs

Similarly, there are several types of graphs.

- Oriented - all edges are oriented
- Un-oriented - all edges are not oriented
- Multigraph - Contains multiple edges

11.3. Terminology

- End vertices (or endpoints) of an edge
- Edges incident on a vertex
- Adjacent vertices
- Degree of a vertex
- Parallel edges
- Self-loop
- Path
    - sequence of alternating vertices and edges
    - begins with a vertex
    - ends with a vertex
    - each edge is preceded and followed by its endpoints
- Simple path
- path such that all its vertices and edges are distinct
- Cycle
    - circular sequence of alternating vertices and edges

    - each edge is preceded and followed by its endpoints
- Simple cycle
    - cycle such that all its vertices and edges are distinct

11.4. ADT Graf supports operations

Access operations

- aVertex()
- incidentEdges(v)
- endVertices(e)
- isDirected(e)
- origin(e)
- destination(e)
- opposite(v,e)
- areAdjecent(v,w)

Update operations

- insertVertex(o)
- insertEdge(v, w, o)
- insertDirectedEdge(v, w, o)
- removeVertex(v)
- removeEdge(e)

General Operations

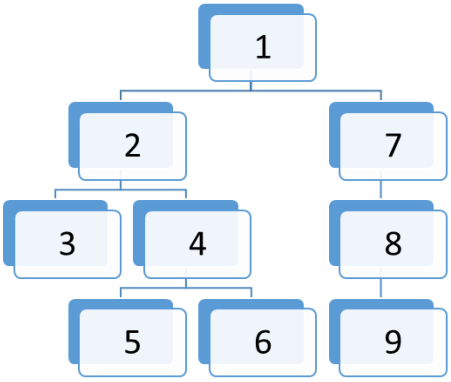
- numVertices()
- numEdges()
- vertices()
- edges()

The graph structure needs to be searched in some way. There are two options - DFS (Depth-First Search) and BFS (Breadth-First Search).

11.5. Depth-First Search

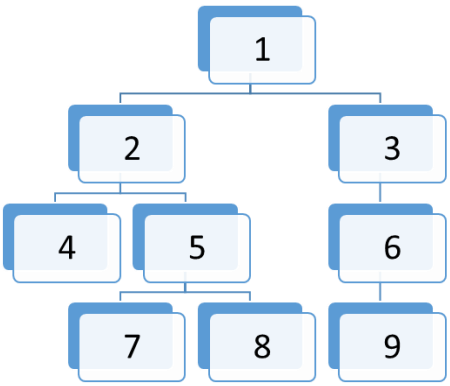
Depth-first search is a complete algorithm (passes through each node). Its principle consists in the fact that it expands the first successor of each peak if it has not visited it yet. If a peak is encountered from which it is not possible to continue (there are no successors or all of them have already been visited), it goes back by backtracking.

It passes the nodes in the following order:



11.6. Breadth-First Search

Breadth-First Search is a similar algorithm, it passes through all the neighbors of the starting peak, then the neighbors of neighbors etc. to pass through the entire connection component



It passes the nodes in the following order:

The fundamental question of graph theory is finding the shortest path between two nodes. There are several algorithms for that.

Dijkstra's algorithm

- non-negative weights on the edges
- $O(|V|^2+|E|)$ – V number of vertices, E number of edges

Bellman-Ford algorithm

- Graph can have negative edges
- $O(V \cdot E)$ – slower than Dijsktra's alg.

Floyd-Warshall algorithm

- Directed/oriented graph with non-negative edges
- It finds the shortest path between all vertices
- Time required – $O(V^3)$, memory required - $O(V^2)$

Johnson's algorithm

- oriented graph, it may contain negative edges
- find the shortest paths between all pairs of vertices in a sparse, edge weighted, directed graph. It allows some of the edge weights to be negative numbers
- $O(V^2 \log_2(V) + VE)$

12. Genetic algorithms

12.1. Explanation

Genetic algorithms belong among the evolutionary algorithms and are parts of artificial intelligence. They are a class of heuristic algorithms.

They use the knowledge of evolutionary biology to solve complex problems for which there is no exact algorithm.

It imitates the techniques of evolutionary biology:

- Heredity
- Mutation
- Natural selection
- Crossover

The principle of genetic algorithms works according to the scheme:

1. Initialization – generate the 0 generation
2. Begining of cycle – Choose (randomly) several individuals from whole population
3. Create a new generation using the following methods:
    i. crossover - „swap“ parts of few individuals
    ii. mutation – randomly change of some genes
    iii. reproduction – copy individuals without changes
4. Calculate the capability of the new generation
5. Termination of the cycle - Repeat from point 2 until the termination condition is reached
6. The end of algorithm – the individual with the best capability is the main algorithm output and represents the best possible solution

12.2. Terminologie

- Phenotype - an individual's designation
- Genotype, genome, chromosome - representation of phenotype
- Chromosome - divided into different linearly arrayed genes (i -th chromosomal gene of the same type representing the same characteristic)
- Alleles - Various gene values
- Fitness value - ranging from 0-1, expresses the quality of each individual

Each individual can be encoded (genetically described) in a different way. The descrip-

tion method can be important for the success or failure of solving a specific task.

12.3. Example:

0th Generation (fitness value # "1"):

- 0100011011 *f*=0,5
- 0101000100 *f*=0,3
- 1010110000 *f*=0,4
- 1110111000 *f*=0,6

Selection

- Weighted roulette: $p_i = \frac{f_i}{\sum_1^N f_i}$
    - Probability of being a parent
- Tournament method
    - Random selection of groups from each parent group becomes the person with the highest fitness value
- Trimming
    - We sort all the individuals according to the f value, cut the low value part, select the parents from the rest
- Random selection
    - The simplest method, f value does not play a role in selecting an individual for parenting
- Crossover
    - Parents exchange parts of their genetic code
    - Simplest method– one point crossover
    - Place for cutting – randomly chosen
        - X: 010001|1011
        - Y: 111011|1000
    - P: 0100011000 *f*=0,3
    - Q: 111011 1011 *f=0,8*
    - More-point crossover , possibility of more than two parents

- Mutation
  - Random change of the random gene in an individual
  - Very low probability

5. 0100011011 ⇒ 0101011011
6. 0101000100 ⇒ 0101100100
7. 1010110000 ⇒ 1010110100
8. 1110111000 ⇒ 1010111000

  - It is possible to reach properties which are not in the original generation

- Termination
  - This generational process is repeated until a termination condition has been reached. Common terminating conditions are:
  - A solution is found that satisfies minimum criteria
  - Fixed number of generations reached
  - Allocated budget (computation time/money) reached
  - The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
  - Manual inspection
  - Combinations of the above

DATABASE

1. BasIc concepts of the database

The database (or data base) is a certain set of information (data), mostly a table with records stored on a storage medium. In the broader sense, the database includes software resources that enable to manipulate and access stored data.

Database is a set of records that are gathered for a specific purpose. We mainly use databases for storing comprehensive information. Database systems are available as a part of office packages (e.g. MS Access, OpenOffice.org Base). These systems are also available as standalone programs that are used to create large databases, for example, MySQL, Oracle and others.

Database is a set of data (information about real-world objects) that is interlinked in some way. Data is an expression for data used to describe a phenomenon or property of the observed object. They represent a form of presentation of real objects (characters, symbols, images, facts, events), thus reflecting the state of reality at a certain point in time.

Information is a message that a certain phenomenon has occurred. It arises from assigning meaning to data and exists in relation to the recipient. It serves to inform about changes in perceived reality. We encounter databases in everyday life very often. We present the common use of large-scale databases:

- database of timetables,
- the state administration database,
- Information systems of banks, schools, offices
- hospital patient recording systems,
- Libraries databases.

1.1. Database management system

Database management system (DBMS -) is software equipment that ensures working with the database, i.e. it creates an interface between application programs and stored data. Sometimes this concept is confused with the concept of a database system. However, the database system is a DBMS along with a database.

Entity

Any object (person, animal, thing or phenomenon) of the real world that is captured in the data model. An entity must be distinguishable from other entities and exist independently of them.

Data

Expression for data used to describe a phenomenon or property of the observed object. Data are obtained by measuring or observing, and can be divided into continuous and attribute data. Continuous data are related to a continuous scale, while attributive data are not.

Information

Information is data that brings us new findings.

Records and attributes

Table rows with attribute values for one object (entity) that must differ from each other. Attributes are columns of the table. Attributes are assigned a specific data type and domain, which is a set of admissible values of the given attribute. The row is cut over the columns of the table and serves to save the data.

Attributes, fields

Objects (entities) properties monitored:

- create table columns
- can acquire various values
- the fields are of a particular data type (number, text, date, ...)

Primary key

It identifies the record (line of the table). It is an attribute (field) that has a unique value for each entity, such as a birth number; usually it is an auxiliary field with a record ID number.

Foreign key

An attribute that is the primary key in another table.

Index

This is a way of arranging a table.

- the order of the records in the table during the "life" of the database does not change; the index helps to find data quickly in the table
- The index creates an auxiliary file with a table arranged by a specific field
- one table can have several indexes arranged by various attributes
- the primary key is always an index

1.2. Database Structure

The most common databases are relational databases. In these, the data are stored in smaller tables to ensure minimal data redundancy. Tables are interconnected using relations. Relations determine relationships between tables and determine the intercon-

nectedness of individual tables. Each of these tables should contain data related to only one type of object (e.g. table of orders, clients, prices, goods, etc.). In order to create a good database, it is necessary first to suggest the appropriate structure of individual tables. These tables must then be interlinked by relations. Tables form the basis of the entire database structure. The basic rules for table design are as follows:

- any information should only be included in the database once,
- each table should contain information about one type of object,
- when designing tables, the future extent of the data should be taken into account.

1.3. Database table

One table should contain information about one type of object. The database table is similar to a regular table. The lines contain records (about one object) and as columns, items or fields are designated. The field sometimes refers to the intersection of a line and a column that contains a single value (data element).

Zaměstnanci

ID_zamestn	Jméno	Příjmení	Datum naro	Pohlaví	Telefonní čí:	ID_funkce
1	Tomáš	Novák	28.4.1980	muž	723 123 456	F_03
2	Josef	Koblížek	15.3.1976	muž	728 452 123	F_01
3	Petra	Maková	5.2.1985	žena	724 556 115	F_02
4	Václav	Sýkorka	30.6.1970	muž		F_06
5	Denisa	Rosolová	12.12.1956	žena		F_05
6	Michal	Aspik	3.6.1976	muž		F_04
7	Dominik	Kokeš	14.2.1981	muž		F_04
8	Tereza	Železná	25.10.1978	žena		F_02
9	Vladimíra	Mimořádná	17.11.1989	žena		F_02
10	Jakub	Pekelník	5.12.1973	muž		F_05

In the above table, the employees create rows of records with information about the individual employees. The columns represent fields where we always see one type of data (text, date, number). Each column (item) has a name, selected data type (e.g text, number, yes / no, date and time) and size. Other features (format, default, etc.) can be assigned. See more about this issue in video guides.

1.4. Primary key

In most cases, we need to identify each entry in the table. The so-called primary key is used for this purpose. A primary key is an field that is intended to ensure unambiguous identification of the individual records in the table. The primary key is usually a single field (so-called simple primary key), but it can also consist of more fields of the tablefield (the so-called composite primary key).

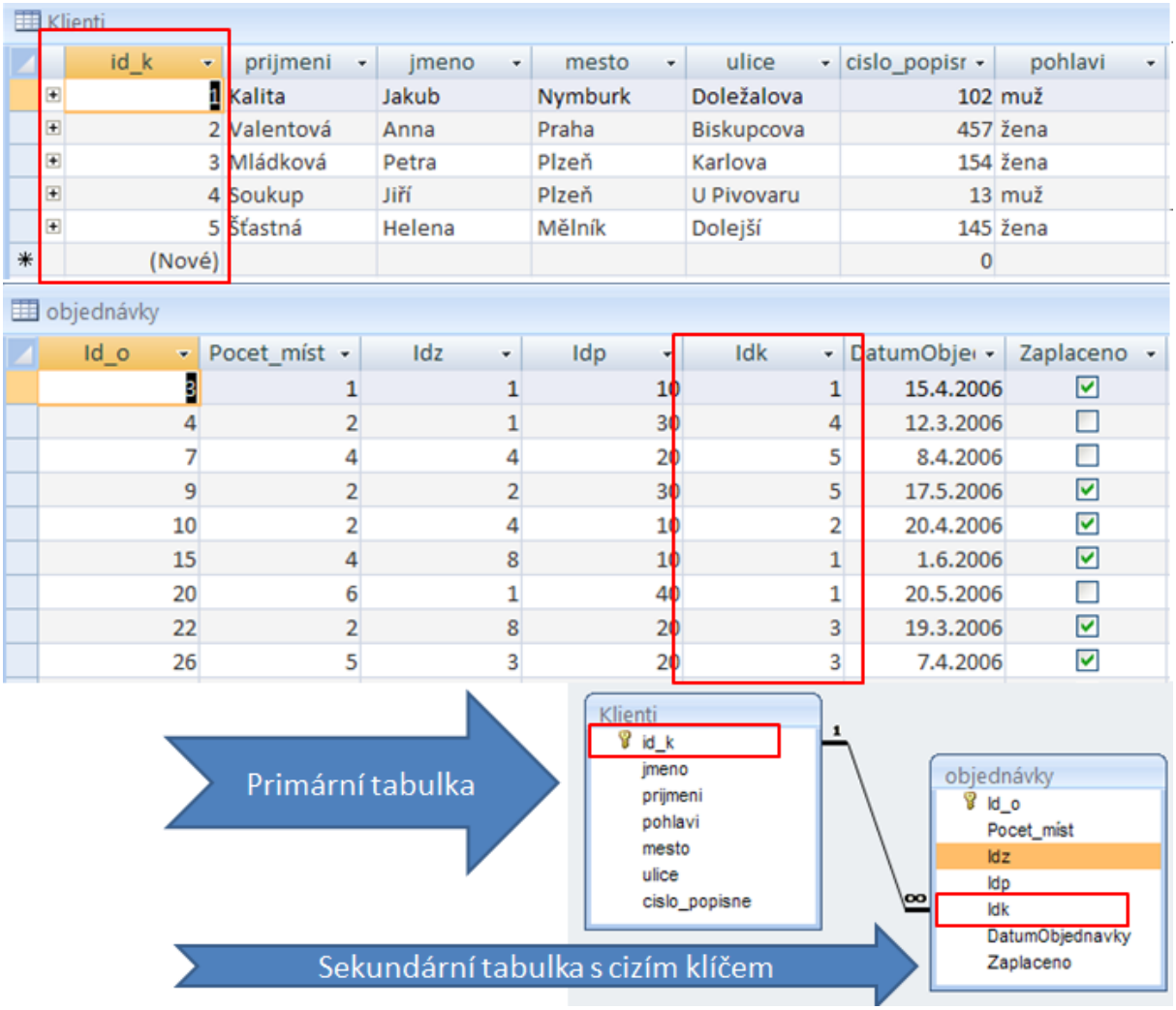
There can be only one primary key in each table. With the primary key, information is searched for more quickly, and relations with other tables are created. The values in the primary key field must be unique for each record. Primary key is one of the indexes. For quicker recordssearch and sorting according to a particular table field it is advisable to use field indexing (indexes). If we need to searchaccording to a different table column than the primary key, we set the index to it. With the index set, sorting, searching and editing the values in the tables is faster.

1.5. Relation

In a relational database, individual tables are linked by relations. The main purpose of relations between tables is to limit the occurrence of redundant data. Data should not be repeated in different tables within a single database. The relation is based on the link of the same values between the unique field (primary key) of one table and the corresponding field of another table. To create a relation between two tables, you must have a special field in each table. In one of these tables this is the primary key. We refer to such a table as the primary table. In the second table, we create a special field for relation purposes. We denote this field as a foreign key, it contains the values of the primary key from another table. A table containing a foreign key is referred to as a secondary table.

1.6. Primary and secondary table

The primary key field contains only unique values. The foreign key field may contain the same values. The primary and foreign key fields must contain the same values for correct relation creation. The figure below shows a primary table that contains individual customer records. The primary key is the first field that contains the customer number. This table is linked to the secondary table, which consists of records of customer orders. The foreign key in the secondary table is the field containing the number of the customer who placed the order. From the example above, we can see that one record in the primary table corresponds to one or more records from the secondary table. In other words, one customer can place one or more orders. This is relation type 1: N.



1.7. Types of relations

We distinguish three basic types of relations:

Relation type 1: 1 (extraordinary) - one record of the primary table corresponds to just one record in the secondary table. For example, one person is assigned just one birth number.

Relationship type 1: N (most common) – One record in the primary table corresponds to one or more records in the secondary table. We have described this case on the above mentioned example with customers and orders.

Relation type M: N (very often) - one or more primary table records correspond to one or more records in the secondary table. We solve these relations using a joining table that we connect with the original tables using two relations of type 1: N.

1.8. Referential integrity

Referential integrity maintains relations between tables. It will not allow us to insert a record in the secondary table that does not have a corresponding record in the primary table. It also monitors the change of the foreign key values when the primary key is changed. Within referential integrity, it is possible to set the rules for deleting records.

1.9. Operation of the database

Professional databases contain a great deal of important information. Persons working with such a database can be divided into several groups:

- **database specialist** designs and creates professional databases,
- **user** inserts data, maintains data and retrieves information from the database,
- **database manager** provides users with access to certain data, is responsible for database recovery after a crash or a fatal error.

2. Database models

From the point of view of storing data and the links between them, we divide databases into the basic types:

2.1. Hierarchical data model

Data are arranged into a tree structure. Each record represents a node in the tree structure, the relations between the records are of the parent / child type. Finding data in a hierarchical database requires navigating through the records to the child, back to the parent, or to the side to the next offspring. The biggest disadvantages of the hierarchical layout are the complexity of inserting and deleting records and, in some cases, the unnatural organization of data.

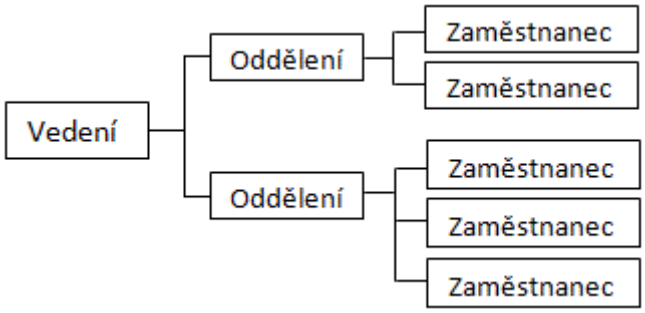


Fig. 1 - Hierarchical model

Legend: vedení – management, oddělení – department, zaměstnanec - employee

2.2. Network Data Model

The network data model is essentially a generalization of a hierarchical model complemented by multiple relations (sets). These sets interconnect records of the same or different type, the connection being made to one or more records. The access to interlinked records is straightforward without further searching; there are operations at disposal: finding the key record, moving the first child in the sub-set, moving the other child in the set, moving up from the child to its parent in another set. The disadvantage of network database is particularly rigidity and a difficult change in its structure.

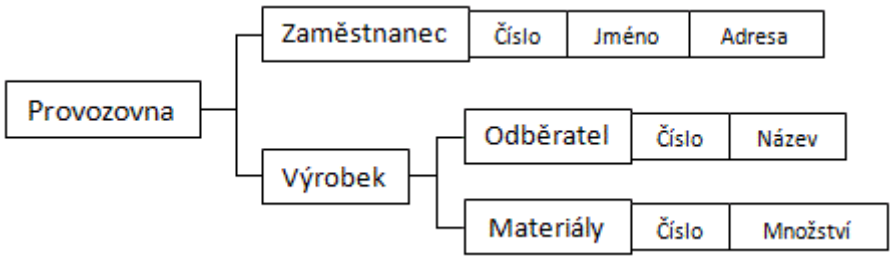


Fig. 2 - Network model

Legend: provozovna – premise, zaměstnanec – employee, výrobek – product, číslo – number, jméno – name, adresa – address, odběratel – customer, název – name, materiály – materials, množství - quantity

2.3. Relational data model

The relational database model is among the youngest and most used ones. At present, it is most often used by DBMS. The model has a simple structure, data are organized in tables that consist of rows and columns. All of these database operations are performed in these tables.

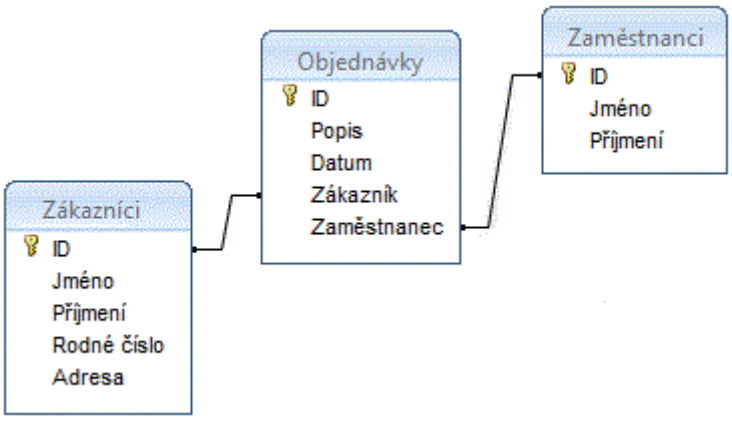


Fig. 3 - Relational model

Legend: zákazníci – customers, Jméno - name, příjmení – surname, rodné číslo – birth number, adresa – address, datum – date, objednávky – orders, popis – description, zaměstnanec - employee

The relational model database must meet the following two criteria:

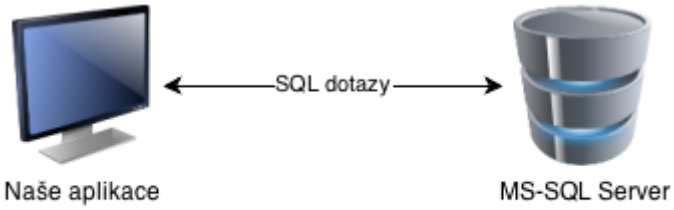
- The database is perceived by the user as a set of relations and nothing else.
- The selection, projection and connection operations are available in the relational DBMS without requiring explicitly predefined access paths to perform these operations.

In addition to relational databases, there are aforementioned object databases. It addresses the problem of object and relational incompatibility. They provide the same comfort as ORM, but internally do not need to convert data into tables, they are stored as objects. In theory, there is no performance or other reason for not replacing relational databases. In practice, however, they are almost unused and we can only hope it will change over time.

2.4. Attached application

You can use the access app when you need to read or change data in real time. Using DataReader, Command and Connection classes, we send SQL statements directly to SQL and get results.

The situation is illustrated in the picture:



2.5. Object databases

Database management systems (DBMS) are currently the most widely used tool for storing and manipulating commercial application data. They provide relatively simple management of large databases, integrity, efficient information search and processing, fault and failure tolerance, simultaneous multi-user access and other benefits.

The Relational Data Model - the most common model used in today's database systems - limits the structure and relations of the stored data to a mere set of tables above a predefined set of basic data types. The undisputed advantage of this model is its simplicity and, as a result, easy standardization and portability. However, the disadvantage is the difference of the real data schema from the internal database table model. All relations between data can only be represented by tables, which, for an application with a more complex data model, leads to a number of tables interconnected by auxiliary links, the so-called keys. This results in a loss of clarity, the database becomes less manageable, and future changes in the application data model force programmers to intervene in their table representation. Relational databases are and will remain the primary means of managing commercial applications that are characterized by a large amount of data with a simple structure. However, many applications, such as design, multimedia, geographic systems, etc., need such a data model that will allow for better correspondence between complex real data and their representations in the database system. This mod-

el is an object data model. The object data model in database systems is based on the principles of object-oriented modeling and programming. However, it is further enriched with techniques of persistence, representation of relationships, questioning, transaction access, etc.

2.6. Fundamentals of Object Orientation. Objects and classes

An object-oriented model is based on the decomposition of real world information into so-called objects. An object is any (even structured) entity that is uniquely and independently identifiable within a certain context of the outside world. Thus, the object has an unambiguous identity, and every two or more similarly identical objects are mutually distinct. The object identity is determined by the object identifier (oid), which is generated by the system, unique, unchanging during the the object lifetime, hidden for both the programmer and the end user. Objects are characterized by means of classes. Class is an abstract description of the object; it determines the data components of the object and the operations (called methods) that can be performed on the object. Each object is an instance of a class, it is possible to instantiate a generally unlimited number of structurally identical objects from one class. in addition, we distinguish the class interface

2.7. Literals

In addition to objects, the concept of literal is introduced within an object-oriented model. A literal is a data entity of a particular data type that, unlike object, does not have its own identity. Literals typically occur as object attributes. The set of operations above the literal data type is fixed, it can not be changed. Objects and literals are related to the concept of mutability. Mutability is perceived as the ability to change data while preserving identity - in this sense, objects are variable because it is possible to change the values of their data components while retaining their original identity. In contrast, literals are not variable.

2.8. Operation

There are several basic types of operations on objects. Each object has one or more constructors. The purpose of the constructor is to initialize the object at the time of its creation. Another part of each object is a destructor. Destructor is called at the moment of object destruction and its purpose is to clean the object before it is removed. An important operation on objects is copying. There are so-called shallow and deep copies. In the case of a shallow copy, the attributes of the object are copied, but any references to other objects point to the same objects as the original object. In contrast, a deep copy will not only copy the attributes of the copied object, but also create copies of the objects to which the original object referred to. Other typical operations are methods for detecting and assigning attribute values, methods performing calculations and manipulation with object attributes, methods producing user output, etc.

An object data model is the data stored in the object structure. It is usually the object's interlayer between the code and the database where there are data the application works with and does not burden the DB server with queries.

Object-relational data model

The object-relational data model is a classical tabular database extended by **abstract data types** (ADT). ADT are user-defined types consisting of the basic data types of the database. Which violates 1NF?

- Object features are now implemented in all major DBMSs.
- Object types and their methods are stored together with data in the database, so the programmer does not have to create similar structures in each application.
- The programmer can access a set of objects as if they were one object.
- Objects can simply represent bindings where one entity consists of other entities (without the need to use bindings).
- Methods are run on the server - there is no inefficient data transmission over the network.

Object Data Types

It may include:

- data - attributes
- operations - methods
- The ORSRBD-specific is that we can see the data from both relational (records, operations) and object point of view (objects, methods).

Types of methods:

- Member methods - are called on a particular object.
- Static methods - are called on the data type.
- Constructor - The default constructor is defined for each object type.

3. Integrity of the database

The integrity of the database means that the data stored are consistent towards the defined rules. Only data that meets the predefined criteria can be entered (for example, respecting the data type set for the given table column, or other limitations permissible for the given column). Integrated constraints serve to ensure integrity. These are tools that prevent incorrect data from being inserted or lost or damaged by existing records while working with the database. For example, it is possible to secure deleting of data that have already lost its meaning - for example, deleting the user, removing the rest of his records in other database tables.

3.1. Types of integrity constraints

Entity integrity constraint - Mandatory integrity constraints that ensure that the primary key of the table is complete (prevents the storage of data that would be the same as in any other row of the table).

Domain integrity constraints - Ensure adherence to the data types / domains defined for the database table columns.

Referential integrity constraints - deal with the relations of two tables, where their relations are determined by the binding of the primary and foreign keys.

Active referential integrity - Defines the activities that the database system performs when some rules are violated.

3.2. Maintaining integrity constraints

In principle, there are three ways to ensure that integrity constraints are respected.

1. Locating simple mechanisms for maintaining integrity constraints on the database server.
    - This is the best way to protect data.
    - However, the user usually has to wait longer for the system response and their portability to another database system cannot always be secured.
2. Locating the protection mechanisms to the client side.
    - the best option for the convenience and independence on the database system,
    - need for control mechanisms for each operation can cause errors for applications, and multiple applications need to be repaired.
3. Separate server-side program modules.
    - in modern database systems, so-called triggers are implemented for this pur-

pose = independent procedures that can be initiated automatically before and after operations manipulating with data.

- This way also enables the implementation of complex integrity constraints,
- The disadvantages are the very limited possibility of transfer data to another database system on the server.

The ideal solution is the combination of the previous ones depending on the specific conditions. Checking integrity constraints is typically performed after each operation, which reduces server requirements. There is no need to record which checks are to be made later. However, more complex integrity constraints cannot always be verified, it is therefore possible to check compliance with the rules only after completing the entire transaction.

3.3. Relations between tables

Relations are used to bind data that are related and located in different database tables. In principle, there are four types of relationships.

- there is no connection between tables, so we do not define any relationship,
- 1: 1 (one record in the table corresponds to just one record in another database table and vice versa),
- 1: N (assigns one record of the table to multiple records of another table),
- This is the most used type of relation because it corresponds to many situations in real life.
- M: N (allows several records from one table to be assigned to several records from the second table).
- for practical reasons, this relationship is most often implemented by combining two relationships 1: N and 1: M which point to the auxiliary, so-called coupling table consisting of a combination of both keys used.

3.4. Normal forms

Normalization means the process of simplifying and optimizing the proposed database table structures. The main objective is to design database tables so that they contain a minimum number of redundant data. The correctness of the structure design can be evaluated by any of the following normal forms.

Zero Normal Form (0NF)

- The table contains at least one column (attribute) that can contain more kinds of values.

First Normal Form (1NF)

- Table columns cannot be further divided into parts bearing some information -> the elements must be atomic.
- One column does not contain composite values.

Second Normal Form (2NF)

- The table contains only columns that are dependent on the whole key.

Third Normal Form (3NF)

- The table is in the third normal form if there are no dependencies between non-primary columns.

Fourth Normal Form (4NF)

- The columns in the table describe only one fact or one connection.

Fifth Normal Form (5NF)

- by adding any new column, the table would split into multiple tables

3.5. Database integrity

The database integrity means that the database complies with the specified rules - integrity constraints. These integrity constraints are part of the definition of the database, and the database management system is responsible for their fulfillment.

Integrity constraints may refer to individual values entered into the database fields (for example, a mark from a subject must range from 1 to 5), or it may be a condition for combining values in some fields of a record (for example, the date of birth must not be later than the date of death). Integrity constraints may also apply to a whole set of records of a given type - this may be a requirement for the uniqueness of the values of a given field or combination of fields within the entire set of records of that type that occur in the database (for example, the ID number in the person records).

Frequently used integrity constraints in relational databases are referential integrity. It is a requirement for a record field to contain a reference to another record somewhere in the database; such a referenced record actually must exist, so that such a link does not lead to "empty" and not a so-called database orphan.

Integrity constraints

It should be ensured that only records corresponding to real-world relationships (such as the age attribute should not get negative values) get into the database. To handle such cases, integrity constraints are used to determine a range of values that a specific attribute can take. Integrity constraints specify what limitations and internal relationships the database data must meet. Relations that meet the integrity constraints are permissible.

Relational database system

- must enable efficient management and analysis of the stored data,
- has to fulfill the following three basic functions:
  - enable to define the data type,
  - enable to work with data,
  - the system should include the means to perform the required activities with data – e.g. sorting data, filtering, calculatins, managing data management,
- In particular, the system checks especially the access to the data, i.e. who is authorized to access the data and who can update it.
  - The system controls data sharing among multiple users.

4. Relational database model

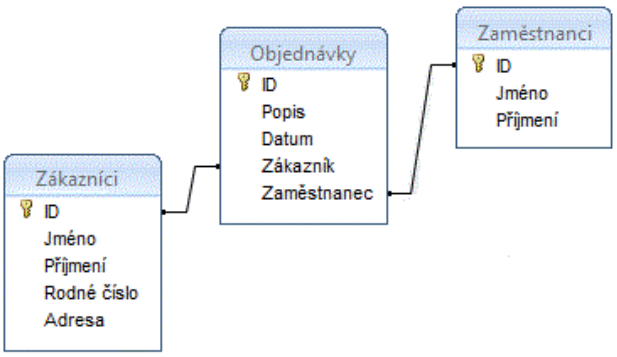


Fig. 3 - Relational model

The relational model database must meet the following two criteria:

- The database is perceived by the user as a set of relations and nothing else.

- At least, selection, projection and connection is possible in the relational DBMS without requiring explicitly predefined access paths to perform these operations.

4.1. 12 rules for relational DBMS:

1. **Information rule:**

All information in the relational database is expressed explicitly at the logical level in a single way - by the values in the tables.

2. **The security rule:**

All data in the relational database are guaranteed to be accessible by combining the table name with the primary key values and the name of the column.

3. **Systematic processing of zero values:**

Zero values are fully supported by the relational DBMS to represent information that is not defined, regardless of the data type.

4. **Dynamic on-line catalog based on relational model:**

The description of the database is expressed at the logical level in the same way as customer data, so an Authorized User can apply the same relational language to his query as a user when working with data.

5. **A comprehensive data sub-language:**

The relational system can support several languages and different modes used in terminal operation. However, there must be at least one command language with a well-defined syntax that supports data definition, view definition, interactive and program data manipulation, integrity constraints, authorized database access, transaction commands, and so on.

6. **View creation rule:**

All views that are theoretically possible are also system-configurable.

7. **Insertion, creation and deletion capability:**

Ability to maintain relational rules for both basic and derived relations is maintained not only when viewing data, but also by data penetrating, adding, and deleting operations.

8. **Physical data independence:**

Application programs are independent of the physical data structure.

9. **Logical data independence:**

Application programs are independent of changes in the logical structure of the database file.

10. Integral independence:

Integral constraints must be defined by relational database resources or its language and must be able to be stored in the catalog and not in the application program.

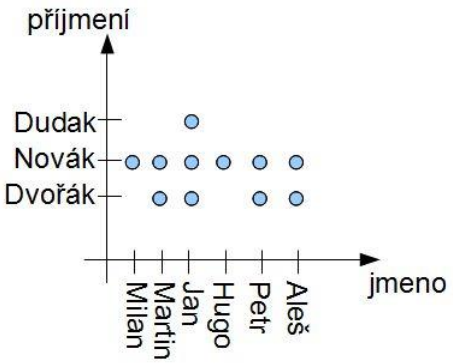
11. Independence of distribution:

Relational DBMS must be able to implement on other computer architectures.

12. Database access rule:

If the relational system has a low level language, then this level can not be used to create integrity constraints, and it is necessary to use a higher level relational language.

4.2. Relational Data Model

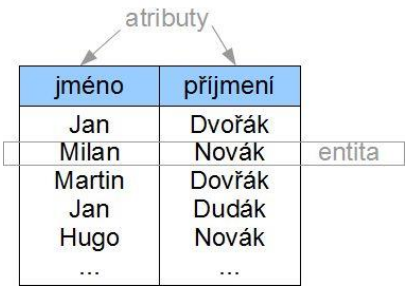


A relational data model is a way of retaining data in tables. It is called Relational because the table is defined over a relation.

Relation is basically a table. It is defined as a subset of the Cartesian product of the domains.

The relation in the figure on the right is therefore a subset of the set of {Dudák, Novák, Dvořák} x {Milan, Martin, Jan, ..., Aleš}

Unlike a mathematical relation, the database one changes over time (by adding and removing relation elements). In addition to basic set operations, there is a selection operation - row selection and projection - column selection for a database relation.



A domain is a set of all the values that an attribute may acquire, iIn other words, the range of attribute values. In practice, the domain is given by integrity constraints (IC).

* Note Figure: Accepting Attribute Domain is a set of {Dudak, Novák, Dvořák}. **Attribute** is an entity property. In terms of table, it is a column.

The relational schema can be seen as a structure of the table (attributes and domains).

Example for a table (relation) Teacher: Attributes: ID, name, surname, function, office

Domains:

- D1 - Three letters from last name, three digits of the order Numbers
- D2 - Name calendar
- D3 - set of surnames
- D4 - set of functions (assistant, scientist, teacher, ...)
- D5 - A101, A102, ... A160

Relational scheme: Teacher (ID, name, address, function, office)

Relation: Teacher = {(nov001, lukas, novak, scientist, A135), (kom123, jan, komensky, teacher, A111)

It is defined as R (A, f), where A is a set of attributes (A1, A2, ... An) and the function f (Ai) = Di assigns domain to attribute.

4.3. Relational Data Model Properties

From the definition of relation the following table properties result:

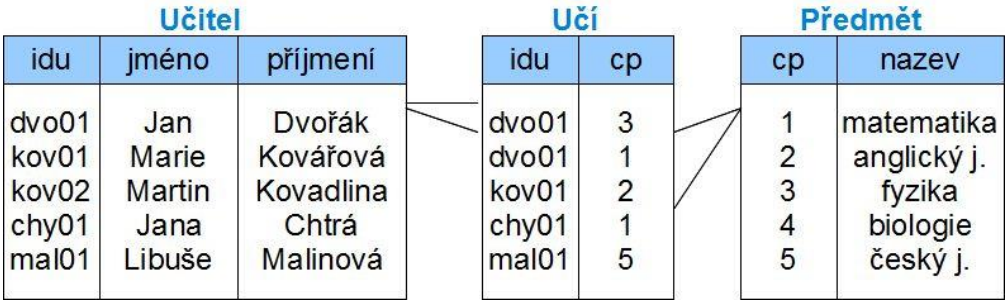
- Column homogeneity (domain elements)
- Each value (the value of the attribute in the column) is an atomic entry
- The order of rows and columns does not matter (they are sets of elements / attributes)
- Each row of a table is uniquely identifiable by the values of one or more attributes (primary key)

4.4. Relationships in the relational model

Generally, bindings in the relational model are implemented using another relation (table). This is the so-called coupling table. It contains the relation attributes (tables involved in the bindings) that uniquely identify their entities - the primary keys. If the table contains an attribute that serves as the primary key in a different table, it contains a foreign key. The coupling table therefore contains foreign keys.

In the figure below, there is the Teacher relation with Primary id Key and the Subject Relation with Primary cp Key. In order to express the relationship Teacher TEACHES The subject a new TEACHES Learning relation was created, which contains two foreign keys (idu referring to the teacher and cp referring to the entity of the subject). Now, we can link the teacher with the subjects using the coupling table, according to who teaches what.

And why wasn´t a subject attribute added to the Teacher table? Simply because the teacher can teach more subjects. Between the teacher and the subject is M: N relationship. So even if we added Teacher attribute to the Subject table, this relationship would not be solved (the subject can be taught by more teachers). For a 1: N relationship, it would be possible, and it is performed in practice. So the coupling table is only necessary for the implementation of the M: N relations.



Učitel

idu	jméno	příjmení
dvo01	Jan	Dvořák
kov01	Marie	Kovářová
kov02	Martin	Kovadlina
chy01	Jana	Chtrá
mal01	Libuše	Malinová

Učí

idu	cp
dvo01	3
dvo01	1
kov01	2
chy01	1
mal01	5

Předmět

cp	nazev
1	matematika
2	anglický j.
3	fyzika
4	biologie
5	český j.

5. SQL Basics

5.1. Introduction to SQL (description and properties)

SQL - Structured Query Language - is a general tool for manipulating, managing, and organizing data stored in a computer database. It is primarily intended for users, although in many ways it is also used by application creators. It is adaptable to any environment.

The name of this tool is brief, but not accurate. SQL is not just a query language, it can also define data, i.e. table structure, fill columns of data table (field) with data, and define relations and organization among data items. It also allows data access control, i.e., granting and removing access at different levels, thereby protecting data from accidental or intentional destruction, unauthorized reading, or manipulation, and it also enables shared data usage and smooth operation when more users access data at a time.

SQL is a specialized programming language that is used in an appropriate environment, either user-friendly or interactively for immediate task-solving (most frequently queries), or its commands are translated into the host language. However, SQL is not a fully-fledged programming language, because in most implementations there are no control program constructs and other required elements that each general programming language should contain. SQL is a standardized tool for working with relational databases. It is not a database system but a differently integrated part of a database management system.

SQL is primarily an interactive query language - it allows you to get answers to very complicated queries almost immediately. It is a non-procedural tool with a data set access and it is a standardized language, it is comprehensible because it understands data in the form of tables, which is easy to understand for users. It works with relational databases in which the user views data as a set of interlaced tables. Each table represents a data set that is arranged in rows (records) and columns (items). The user refers to the data value as an element in the matrix.

In the majority of cases, the result of a task described in SQL is a data set from one or more tables, a so-called result table, which is not always the final product. It can serve as a set of input data for further processing, e.g. for printing labels or drawing a charts, etc.

SQL can serve as "common language," especially when operating in networks where different database products are used.

The SQL language is also used to create previews (queries). SQL views allow you to create different views of table structures and data for different users. Every user sees only the data they are supposed to see. The data are seen by the user a as a simple table, although in fact the data come from different tables. The data displayed in the views are

dynamic, that is, if the data in the tables (database files) change, the data that display the preview will also change and a vice versa. When working with an update preview (a special type of preview that allows not only data to be seen but also to be added and updated) and the data change, the changes will be reflected in the appropriate tables (database files).

A key term in SQL is a command. Each command begins with a keyword. The word indicates what activity the order is performing. The keyword is followed by one or more optional clauses that specify the nature of the activity being performed, or specify the data with which the order is to work.

Each clause begins with a key word, such as FROM or WHERE. Some clauses are mandatory, others are optional. Each SQL implementation uses its own clauses, in addition to the standard clauses given by the ANSI / ISO conventions, sometimes the function of standard clauses is more or less different or different.

SQL object names contain 1-18 characters in the standard, the first character of which must be a letter, and the name must not contain spaces or punctuation marks. When naming, it is sometimes necessary to qualify the name if the names are the same, and it is not clear what object the name refers to. Name qualification is done using the '.' (Dot) qualifier. Often qualifications are used with table names:

```
<Owner>. <Table_name>
```

or when qualifying columns, which is very common in database systems:

`<Tablename>. <Columnname>`, generally the qualifiers are also allowed in SQL.

Multilevel:

```
<Owner>. <Table_name>. <Column_name>
```

5.2. SQL queries

An introduction to a statements in SQL databases.

Introduction to SQL commands

The basics are to introduce basic SQL statements in simple examples. The information can then be used when programming in Visual Basic, Delphi, Web site programming. The basic introduction to what the database is is mentioned in the a Introduction article. If you already know the database theory, you can see how to create a database in Microsoft Access.

List of SQL statements

The most important commands are: SELECT, INSERT, DELETE, CREATE, FROM, WHERE and many others. But first we will make a simple table on which we will introduce the commands (to make it clearer what the command is doing).

Create a test table

zamestanci : Tabulka

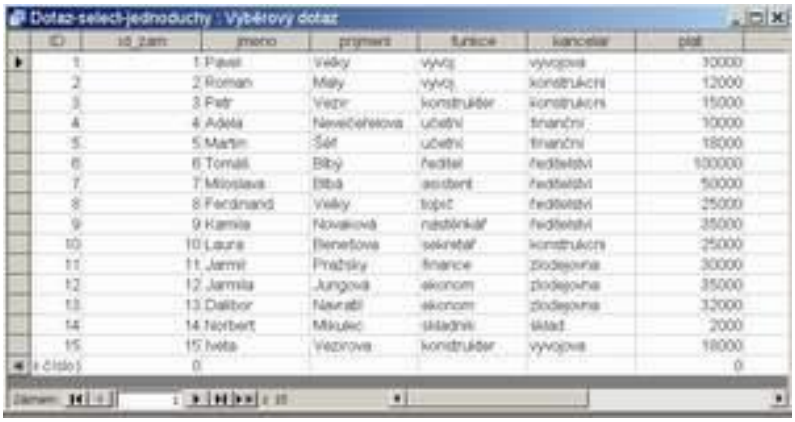
ID	id_zam	jmeno	prijmeni	funkce	kancelar	plat	vek
1	1	Pavel	Velky	vyvoj	vyvojova	10000	20
2	2	Roman	Maly	vyvoj	konstrukcni	12000	25
3	3	Petr	Vezir	konstrukter	konstrukcni	15000	35
4	4	Adela	Nevečeřelova	učetní	finanční	10000	20
5	5	Martin	Šéf	učetní	finanční	18000	23
6	6	Tomáš	Bílý	ředitel	ředitelství	100000	35
7	7	Miloslava	Bílá	asistent	ředitelství	50000	16
8	8	Ferdinand	Velky	topič	ředitelství	25000	66
9	9	Kamila	Novaková	nástěnkář	ředitelství	35000	25
10	10	Laura	Benešova	sekretař	konstrukcni	25000	55
11	11	Jarmil	Pražsky	finance	zlodejovna	30000	36
12	12	Jarmila	Jungová	ekonom	zlodejovna	35000	28
13	13	Dalibor	Navratil	ekonom	zlodejovna	32000	19
14	14	Norbert	Mikulec	skladnik	sklad	2000	48
15	15	Iveta	Vezirova	konstrukter	vyvojova	18000	47
(Nové číslo)	0					0	0

Záznam: 1 z 15

In order to clarify the terms of SQL language, it's a good idea to test it on some simple tables. Once they are fully understood, we can put them on a 10,000-line table. This table can be created in MySQL, FoxPro, Access or some other database program.

SELECT command

An important command. We use the created table and test the application for this command. The simplest form of the statement is this:




```
SELECT *
FROM employees;
```

Shows the contents of the entire table "Employees"

SELECT the name

```
SELECT the name
FROM employees;
```

FROM employees;

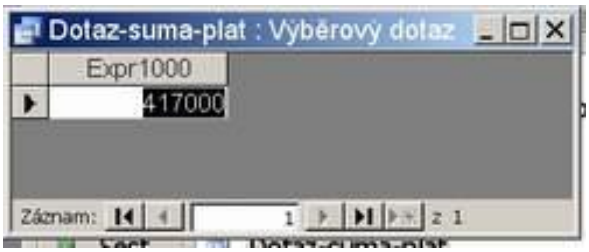
Shows the contents of the "Name" column from the Employees table


```
SELECT DISTINCT function
FROM employees;
```

Shows the contents of the "function" column in the Employees table, but displays only

items without duplicates (so even if there are 4 developers listed only one will show)

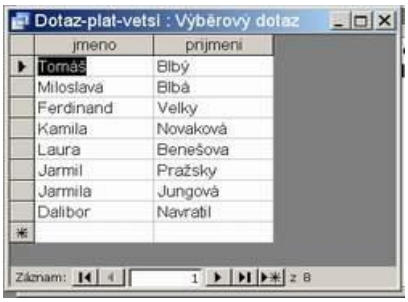
```
SELECT AVG(plat)
FROM zamestanci;

SELECT SUM (flat)
FROM employees;
```



AVG (column) calculates the average from the given column, SUM (column) calculates the sum of the values in the given column, or we can perform calculations in the query itself.

```
SELECT SUM (PLAT + 500)
FROM employees;
WHERE
```

When you have 10,000 lines you will want to limit the calculations to some criteria. This can be accomplished by WHERE. Together with some mathematical symbols (<,>,<=,>=, ..)




```
SELECT name
FROM employees
WHERE pay> 20000;
```

It shows names of employees whose salary exceeds 20,000 CZK

Common simple query syntax

SELECT the list of columns from the WHERE table of the WHERE search terms to ORDER by the sorting columns.

The column list is either a comma-separated list of columns, or a * character to select all columns. The column name is either the column name only or the column-name.class_name. You can also use a table. * to select all columns from the table. The table can be any of:

- Relations (real database table)
- Result of another SELECT query
- View
- result of the JOIN operator (virtual table).

Search term is a condition that must be met for each record that is written in query results. If we want to write a record, first it must t exist in the source table. Therefore, WHERE is a limiting condition. If it is not given, all the rows in the table are written. Sorting columns is a list of comma-separated columns according to which the resulting table will be sorted. The first column is the primary sorting criterion, the second column is a secondary sort - if you cannot decide according to the first column.

Example 1

Enter: Select all Radka and Tomas from the database.

Solution 1

```
SELECT name, surname, nickname FROM person
WHERE name = ’Radek’ ORname = ’Tomas’
```

We can use common logical operators AND and OR, in most databases it can also be written as && and ||. You can use common = to compare strings, but you need to take a few things into account. Depending on the server settings, the letter size does or does not matter. The original (and therefore more frequent) default setting is that the letter size does not matter. In Akela.mendelu.cz server the letter size matters. Additionally, depending on the collation server (and database) setting, diacritics does or does not matterIn Akela.mendelu.cz diacritics matters, so the name must be written exactly as it is stored in the database.

Solution 2

SELECT name, surname, nickname FROM persons WHERE name IN ('Radek', 'Tomas')

An alternative to the previous query is to use the IN operator. IN is a set operator that tests whether the value is in the specified set, so for each row the value of the name column is tested if it is in the set ('Radek', 'Tomas'). The set can be defined either by a direct enumeration of values or by another SQL query.

Example 2

Task: Select all the people who live in Prague from the database.

Solution - step 1

```
SELECT id_address, FROM FROM WHERE address mesto = ’Praha’
```

In the first step we find all the addresses that are in Prague. It is now appropriate to recall the database schema from which we find that there is an id_address column in the person table that is related to the id_address column in the address table. The hypothesis is that if we select all the persons whose address is in the result of the above query we will get exactly those persons whose address is in Prague.

Solution - step 2

```
SELECT name, receive, nickname FROM people
WHERE id_addresses IN ()

SELECT DISTINCT id_address FROM WHERE address
mesto = ’Praha’ ()
```

Now there are two SELECT constructs in the query. The second select is the so-called sub-query, and it is a slightly modified query created in the first step. The first adjustment is that it now only selects the id_address column. This is crucial, given that the parent SQL query looks for id_addresses, so the sub query has to return only the id_addresses. In addition, the IN operator only works with a scalar set and therefore a sub-query (to define a set for the IN operator) must always return just one column. If you use multiple columns in the sub query, the DB server will receive an error: ERROR: subquery has too many columns. The second change is to add the DISTINCT keyword to ensure that the return values are unique (see the following example). In this particular case, it is unnecessary (because the id_address is always unique), but because IN can behave very strangely with a sub-query in which the values are repeated (then it is not a set) as well as a precautionary universal measure DISTINCT is good.

6. SQL - more complex queries

Exercises - SQL queries with multiple tables

6.1. Example 1

Task: write a SQL query that selects all people from the database who have an ICQ number; Select name, surname and ICQ number of the person; Sort by last name in ascending order.

Alternative 1

The easiest - the value of the id_type_contact = 1 corresponds to the ICQ type in the contact_type table. The JOIN operator always works with two tables. The coupling condition is always the equality of values in some columns. The join condition must contain both tables that are in the JOIN section. The join condition must contain columns that link the tables. There is no reference to the contacts table in the person table. The contacts table is a single reference to the person table - the id_object column. INNER JOIN only selects from the table those records that meet the connection condition - that is, only those who have a contact (but because the search term is specified in the query, in this particular case it is also possible to use LEFT and RIGHT JOIN, but it is only a coincidence).

```
SELECT person.name, person.name, contact.contact FROM
      INNER JOIN contacts
            ON contacts.id_osoby = person.id_osoby
WHERE contacts.id_types_contact = '1'
ORDER BY by ASC
```

Alternative 2

It is the same as the previous one but uses USING. If the join condition is the equality of the equally named columns (id_id), its input can be simplified.

```
SELECT person.name, person.name, contact.contact FROM
      Persons JOIN contacts USING (id_persons)
WHERE contacts.id_types_contact = '1'
ORDER BY by ASC
```

Alternative 3

A better one - selects contacts by contact type name, so it does not rely on any id_type_contact value. The contact type name is in the contact_type table. It is important

to think that the result of joining tables is a table and that the JOIN operator always works with two tables.

```
SELECT person.name, person.name, contact.contact FROM
      (INNER JOIN contacts
            ON contacts.id_osoby = person.id_osoby)
                  INNER JOIN contact_types
                        ON contact_types.id_type_contact =
                        contacts.id_types_contact
WHERE contact_name.nazev = 'icq'
ORDER BY by ASC
```

SELECT is a command that can have many other words, it can combine several tables into each other, etc. But the full base of the syntax is: SELECT, then the list of columns we want to get, or aggregate or other functions FROM and the table name from which we want Date get, WHERE, and the expression with the limitations that must apply to the rows you want. In our case, we want the column of the city to contain exactly the value of "Chomutov".

With the aggregate function it is for example:- we want to find out how many rows, that is people are from Chomutov:

```
SELECT COUNT (*) FROM lide WHERE mesto = "Chomutov";
```

It will return one line in one column and it will contain number 2.

The conditions can be much more complicated with boolean AND and OR operators, brackets and SQL functions, the list of which is also in the documentation. Finding people from Chomutov over the age of 30 would be as follows:

```
SELECT * FROM lide WHERE mesto = "Chomutov" AND age> 30;
```

An asterisk means "all columns".

Linking other tables to a SELECT query is done using JOIN. This topic requires more than one table and some relationship between them. For example, the table people, where each person has a unique id, and then a table of ideas with a person_id column and an idea. In the person_id column, numbers would be people who invented the idea, and in the idea column there would be the text of the idea. Then we could make a query that would list all the ideas and add to each one a column with the name of the person. But details go beyond this introduction:

```
SELECT napady.napad, lide.jmeno FROM napady
```

```
LEFT JOIN lide ON lide.id = napady.clovek_id;
```

The result will be a table of ideas where the first column will be the idea and the second the name of the person who invented the idea.

When you insert "DELETE" instead of the "SELECT columns," we have a query for deleting rows. It works just like SELECT (there is a WHERE condition), but instead of getting the results, it deletes the lines corresponding to the condition. To delete Adam, just enter:

```
DELETE FROM lide WHERE name = ”Adam”;
```

Another one from the most frequently used SQL queries is UPDATE. This will edit the row. If Catherine becomes one year older, we will update it as follows:

```
UPDATE people SET age = 30 WHERE name = ”Catherine”;
```

But we can also use mathematical expression and references to existing column values, that is, to increase the age column by one can be done as follows:

```
UPDATE people SET age = age + 1 WHERE name = ”Catherine”;
```

These are all of the most commonly used SQL queries (clauses). Individual queries may have more words for more complex and more precise queries. For their understanding, however, it is necessary to study more individual documentation or to look for a specific problem. For example, when using the aggregate function in SELECT, we get the result of aggregating the entire table. But if we want for example to get temperature averages in specific years and we have temperatures in months, we have to use GROUP BY and when we type in google group by years datetime mysql, we find that we need DATETIME YEAR:

```
GROUP BY YEAR (record_date)
```

In addition to GROUP BY, select has also ORDER BY command, where you specify which columns to use to sort the results and whether to sort them in descending or ascending order. Sometimes the word order is important. For SELECT it is again in the documentation and we can see that first we have to have GROUP BY ... and then ORDER BY ....

7. Graph databases

Graph is a data structure consisting of vertices and edges. The graph database is a data storage and processing system in the form of a graph. In many cases, domain modeling using graph is very natural - such domains include human relationships, relation between genes and proteins, mobile networks, distribution networks of different kinds, neural networks, or ecological networks that capture interactions between organisms.

Graph databases often include different systems that manage data in the form of graphs. Based on such a definition, it would be theoretically possible to see relational databases, or their superstructures as graph databases. Relational databases, however, do not allow the effective storage and querying of graph data.

7.1. Relational databases

Moving through the graph in a relational database requires joins, and it is therefore inefficient for deep passing. One join using index requires a logarithmic time O (log n), without the index it is O (n log n).

For example, human relationships can be modeled using two tables: Man and Relationship. For each passage through the relationship between two people, we generally need one join. Traditional relational databases are optimized for join units. As the number of join moves to tens, relational databases are getting into trouble despite the use of indexes.

7.2. True graph databases

For the purposes of this article, true graph databases are such databases that require only a constant time O (1) for the passage of the graph edge.

True graph databases, like other NoSQL databases, store data in a denormalized (already "joined") form. For true graph databases, you have to pay especially for writing, while querying (reading) is cheaper.

An example of a true graph database is Neo4j, which has been under development for more than a decade and is sufficiently sophisticated to be used for production, or, for example, for the newer OrientDB system.

Examples of databases that are able to store and query graph data but not necessarily efficiently browse the graphs are most triplestores (Sesame, Jena, Virtuoso) or FlockDB (a Twitter project).

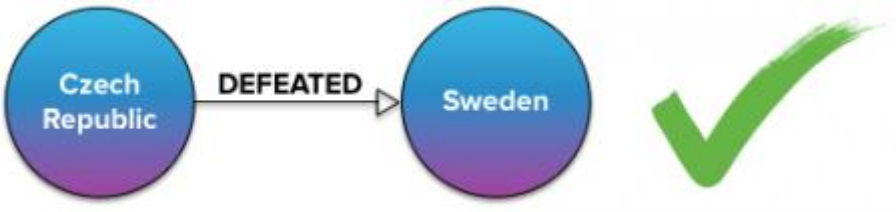
As in the case of most NoSQL databases, there is no single query language for graph databases. However, many graph databases implement Gremlin language developed in collaboration with Neo4j authors.

Transition from the relational world to the world of graphs requires a shift in thinking and viewing data. Although graphs are often much more intuitive than spreadsheets, I've seen persistent mistakes at people who start with graph modeling. In this article we look at one of the most common mistakes - modeling two-way relationships, and finally we will show a real example in which we will continue.

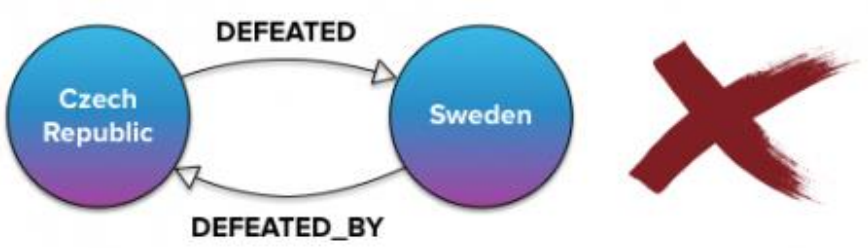
7.3. One-way relationships

Relationships in Neo4j must be of some type that gives the relationship its semantic meaning and also has a determined direction. This direction expresses meaning among entities. In other words, without determined direction, the relationship is ambiguous. .

For example, the following graph shows that the Czech Republic defeated (DEFEATED) Sweden in an ice hockey match. If the direction of the relationship changed, the Swedes would be much happier. Without the direction, we have no clue who the winner is, and that is why the relationship is ambiguous.



Note that from the existence of this relationship it results relationship in the opposite direction, as showed in the graph below. This is a very common case. Another example: Pulp Fiction was directed (DIRECTED) by Quentin Tarantino also means that Quentin Tarantino is the director of Pulp Fiction (IS_DIRECTOR_OF). And this could result in many pair relationships.

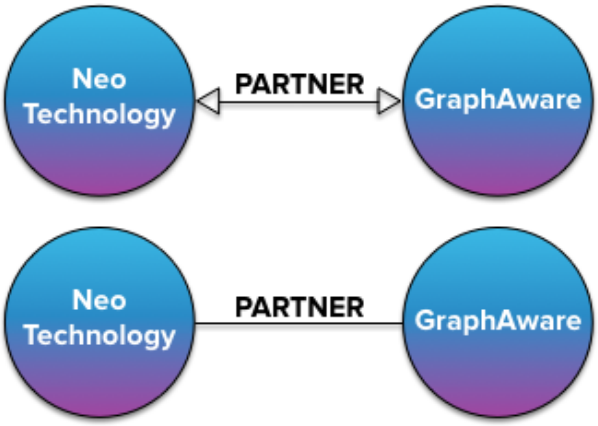


It is this mistake that people often make in graph modeling in the Neo4j and create bi-directional relationships. Since one relationship expresses the second one(symmetric relation), it is uneconomical both in terms of space and moving through the graph (traversing). Neo4j allows you to traverse the relationship in both directions, so also in the opposite direction of the edge. Moreover, due to the way Neo4j stores data, the speed of passing does not depend on its direction.

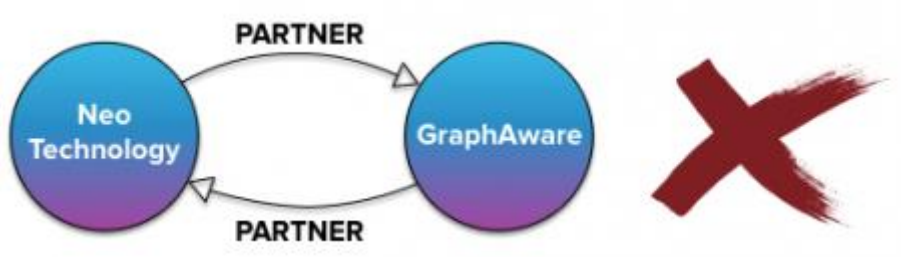
7.4. Two-way relationships

Some relationships are naturally two-way. A classic example is Facebook or a friendship. The relationship is mutual - when you´re somebody´s friend, so he´s a friends of yours (probably). Depending on how we see the graph model, we might say that it is bidirectional or nonoriented.

An example of GraphAware and NeoTechnology are partner companies. Since it is a relationship, we could model a two-way or non-oriented relationship.



But this is not possible in Neo4j - each relationship must have an initial and a final node. Beginners often tend to work with the following model, which has exactly the same problem as the aforementioned ice hockey model with redundancy.



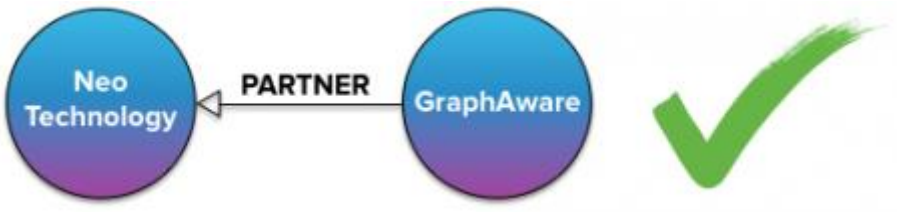
The Neo4j API allows developers to completely ignore the directivity of the relationships when writing queries if they want to. For example, in Cypher, the search for all partner companies with NeoTechnology could be as follows:

```
MATCH (neo) - [: PARTNER] - (partner)
```

The result would be the same as merging the results from these two different queries:

```
MATCH (neo) - [: PARTNER] -> (partner) and MATCH (neo)
<- [: PARTNER]
```

Therefore, the right (or at least the most efficient) way of modeling partner relationships is using a single PARTNER relationship with determining any direction.


Neo
Technology
PARTNER
GraphAware

7.5. Summary

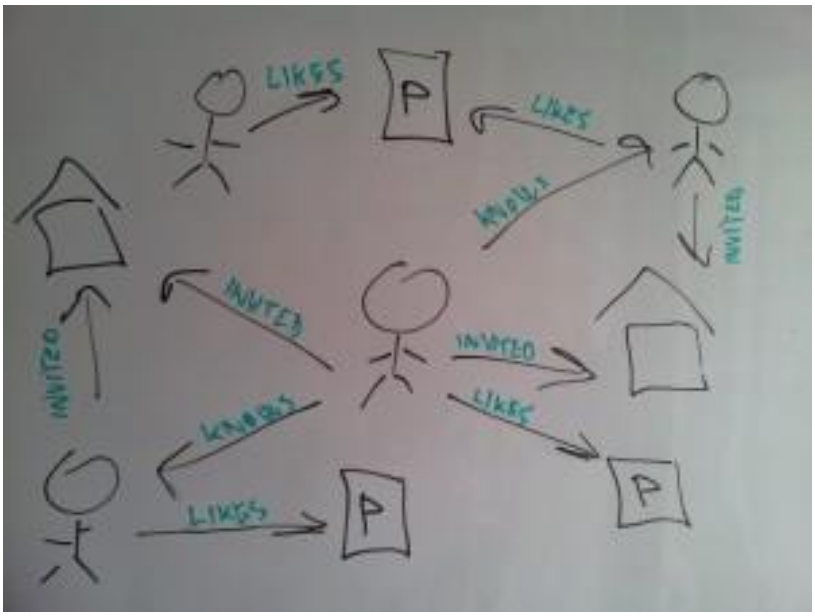
Relationships in Neo4j can be traversed in both directions at the same speed. In addition, directivity can be completely ignored. For this reason, it is not necessary to create two different relationships between entities, if the edges of the opposite direction have the same meaning.

7.6. Graphic Database Modeling Methodology

In the following articles, we will deal with the design, implementation and testing of the Neo4j graph database on the BestPartyToday project (this is a worldwide partylist that offers detailed statistics on events). Currently, the MySQL database uses tens of millions of records, so it will be necessary to move data from a relational database store to a graph one. However, now we will only model the graph database from selected tables and attributes of the current structure of the MySQL database.

The conceptual design of the graph database is referred to as Whiteboard friendliness. In order to quickly reflect all the ideas and insights that were raised during the design brainstorming, a flipchart will do. The resulting layout is very simple to present to other project groups (retail department, account managers, sales managers, etc.) who do not have technical education. Understanding the design is no longer an obstacle for them, as the layout can be easily adapted to real life.

7.7. Implementation of the conceptual model



The photograph of the final design of graph database for the BestPartyToday projet shows five most important tables of the current relational database, represented by icons depicting the nodes and prolationships represented by directional arrows. Through the entities (nodes and relationships) we will traverse the graph and obtain the required data that will be processed and displayed to the user of the application. Their meaning is described in the table below.

8. NoSql database

NoSQL database as the main topic of this work are given most attention to First, it is important to specify what the NoSQL database is. Next chapter is dedicated to CAP Theorem, which explains that one of the reasons why there are so many NoSQL products is that each of them can only provide certain properties at the expense of others, and then the user must decide which combination of properties is the most important for them and then to choose the product accordingly. The lecture focuses also on scalability, i.e. the capability to increase the performance of the database server, either by improving the server using more efficient hardware or by spreading the database to multiple servers. Besides scalability, we also need to mention sharding, a technique that divides databases into multiple parts that can work independently and faster than a whole. Next, the best-known NoSQL products will be mentioned, from which three representatives will be selected to make a detailed description. As already mentioned in the introduction, NoSQL databases are those that do not address the relational way of data control. They are not primarily based on tables and do not use SQL to work with data. Their realm is high search optimization at the expense of small functionality, which is often limited to a simple data storage. These deficiencies, compared to SQL, are offset by scalability and performance in certain data models. NoSQL is suitable for storing large volumes of data that do not need to be interrelated. Structured data are, however, allowed. For transactions, NoSQL databases cannot offer full ACID support, but only eventual consistency. This is a situation where, due to neglecting ACID, we gain more accessibility and better scalability. This approach without "strong consistency" is suitable for NoSQL, which often apply it too. NoSQL has a distributed architecture that is fault-tolerant. Some data may be located on several servers, so the failure of one can be tolerated. These databases usually scale horizontally and manage large volumes of data for which performance is more important than consistency.

NoSQL is literally a combination of two words: No and SQL. It's a technology that stands against SQL. The abbreviation may be confusing and there is no consensus on its meaning. However, the most widespread opinion is that "Not only SQL" is an acronym. Whatever the abbreviation means, NoSQL is now the umbrella name for all databases that do not followRDBMS (relational database management system) but use their own way, often associated with large volumes of data.

What is it?

First of all, it is important to specify what NoSQL is NOT - it is definitely NO SQL, that is, rejecting relational databases. NoSQL means Not Only SQL, meaning that the relational database is not the only way to solve the persistence, that there are alternatives that may in some cases be more appropriate.

For larger applications, we can see that a relational database is suitable for some data, while for other NoSQL databases are more suitable This pragmatic approach when more databases are used in one project is often referred to as polyglot persistence.

NoSQL databases have originated as a solution to real problems - it is not a crazy idea of academics living at a distance from everyday reality, but who have been torn apart from reality but a highly practical matter. The emergence of many of NoSQL databases is related to projects that have had to deal with a huge amount of data - Facebook (Cassandra), Google (BigTable), Amazon (Dynamo), LinkedIN (Voldemort).

The use of a NoSQL database can be meaningful even if we have less data (which a relational databases would be able to process) - because some NoSQL databases offer a data model that is more suitable for our application.

But back to the introductory question. NoSQL Database is data persistence software that is an alternative to classical relational databases and it definitely has its place on market.

9. Transaction

Transaction is a logical unit of work consisting of one or more SQL statements that are atomic in terms of recovering from SQL transaction errors. SQL transaction automatically begins with the SQL initialization command (SELECT, INSERT). Changes made by one transaction are not visible to other competing transactions unless the transaction is not completed.

There are four ways to complete a transaction:

- COMMIT completes the transaction successfully and the changes are permanently recorded
- ROLLBACK interrupts the transaction and all changes are cancelled, the database returns to the pre-transaction state
- within the program - if the program ends successfully, SQL transaction
- within the program - if the program ends with an error, the SQL transaction is cancelled

SQL Access Database Management defines two commands to access table:

- GRANT and REVOKE. The security mechanism is based on - Authorization Identifiers - Ownership – Privileges.

9.1. Solution: Transactions

Transactions can be understood as a sequence of database operations that meet ACID:

1. **Atomicity** - Within a transaction, all operations or neither of them are executed.
2. **Consistency** - The database is consistent before and after the transaction. All integrity constraints must be met.
3. **Isolation** - Individual transactions are isolated. Changes made during the processing of the transaction are not visible in other transactions.
4. **Durability** - After successful completion of the transaction, the data are stored and cannot be lost.

9.2. Transactions in SQL

In SQL, there are the following commands to deal with transactions:

BEGIN	startet eine Transaktion ... befiehlt innerhalb einer Transaktion etwas ...
COMMIT	speichert und beendet die Transaktion

BEGIN	… befiehlt innerhalb einer Transaktion etwas …
ROLLBACK	kehrt Veränderungen an einer Transaktion um und unterbindet diese

Isolation levels

ACID requirements are relatively strong and time-consuming. Therefore, these conditions can be reduced by so-called isolation levels. There are 4 levels (from the weakest to the strongest):

1. READ UNCOMMITTED

The transaction can read data recorded by another transaction without this other transaction being completed with COMMIT. Reading such data is referred to as dirty read. Such data may be inconsistent (time goes down from top to bottom):

2. READ COMMITTED

At this isolation level, dirty read cannot occur. The data we read are always recorded by a transaction that has been successfully terminated with COMMIT. However, non-repeatable read can occur. If we read data more times within a transaction, it can happen that within repeated readings not the same result appears every time. It is possible that between the individual data readings within a transaction, a parallel transaction has been completed successfully, which had recorded / edited / deleted some data.

3. REPEATABLE READ

At this level, it is ensured that there is no non-repeatable read. The data we read once cannot be changed during repeated reading. However, phantom read can occur. If we read data in a transaction more times, it may happen that the repeated reading result will contain new rows that did not appear in the result of the previous reading. It is possible that between the individual data readings within a transaction, a parallel transaction has been completed successfully that has recorded new data.

4. SERIALIZABLE

ACID enforces no above mentioned phenomenon can occur.

10. Procedures and functions, triggers and sequences

Functions, methods and procedures - what's the difference between them? Functions, methods, and procedures are fairly similar to each other - it's the DEFINED sequence of commands, a part of the program that can be repeatedly called for from other parts of the program.

10.1. Formal view

From the formal point of view, we follow the nomenclature of the given programming language.

- Function is a term of functional programming, which occurs in languages like JavaScript or Haskell.
- Functions in Object-Oriented Programming (OOP) are referred to as methods, where the data structures and functional code are linked to objects. These methods can be found in Java, Smalltalk, etc.
- Procedure is a term known from procedural languages. They can be founde.g. in Pascal and in some database systems - so-called stored procedures.

However, we can see functions, methods and procedures from a factual point of view (according to their properties and meaning) and deviate from the terminology of a particular programming language.

Function

Functions in programming are very close to the functions as we understand them in mathematics. Function has a certain definition field (type of input parameters) and a certain range of values (type of return value).

Example of a simple JavaScript function:

```
Function sum (a, b) {return a + b};
```

In object languages like Java, there methods, not functions, but it does not prevent us from writing functions in them:

```
Public static int sum (int a, int b) {return a + b};
```

Although sum () is formally a method (public and static), it is actually a function and the class here only plays the role of the namespace (together with the name of the package) and we do not create its instances (or not). This is a design pattern Library Class.

In Java, such functions are for example in the java.lang.Math class. The following function returns the larger of two numbers:

```
Int x = java.lang.Math.max (a, b);
```

Procedure

The procedure is a special case of a function - it does NOT have a return value and does not have to have even input parameters. They are often used in batch processing - for example, every hour we call the procedure that processes the orders that have accumulated in the database and passes them to another system.

An example of a very simple procedure in PostgreSQL that sums the values of columns a and b to the unaddressed rows and stores the result in column c:

```
CREATE OR REPLACE FUNCTION add ()
RETURNS void AS
$ BODY $
UPDATE table of summaries
SET c = a + b
WHERE c is NULL
$ BODY $
LANGUAGE sql VOLATILE;
```

This procedure is written in SQL - besides, we can write in plpgsql, which allows us to create very complex procedures, or we can use C language or some other.

The above mentioned procedure (formally function) has no input parameters or return value, it is simply a sequence of commands of a language we have named and saved.

It is interesting that the procedures can have parameters - not only the classic input parameters but also the output (or input / output) ones. The output parameter procedure use is similar to function - although it does not have a classical return value. For example, we can pass some data to the procedure, and it will modify it.

We can also simulate these procedures in Java.

A structure:

```
Public class Person {public String name;
Public String Surname; Public String fullname};
```

A procedure:

```
Public class StoredProcedures {public void reportNameName
```

```
(Person o) {o.celeName = o.name + ”” + oName}};
```

We will pass the data (the person) to the procedure and make the required adjustment. However, if you are programming this way in Java, you are probably doing something wrong and do deprive yourself of the main benefits of PPE.

Note: beware of value transfer - if you assign a new person to the variable within the Java procedure, this change will not be reflected in the procedure - the original person remains untouched. But we can work with the attributes of the original person - in this case, the code of the procedure and the code that called it "sees" the same person's instance.

10.2. Method

Methods are part of the class and (if they are not static) are closely related to the given class instance (object). Therefore, they are not functions working with any global variables and data, but they have access to variables of a given instance (in this case, individuals).

The previous example can be overwritten "more objectively" as follows:

```
Public class Person {public String name; Public
String Surname; Public String getCelé Name ()
{return name + ”” + surname}};
```

We do not need a class with stored procedures and move the required functionality closer to the data (to the Person class). Note that we do not have to save even the calculated value fullName - we will calculate it in case somebody needs it, i.e. when somebody calls the getNameName () method.

What is a trigger?

Trigger is a procedure that is automatically triggered when something happens. It can be defined for both DML (modification) and DDL (creation) operations. Plus, it can be said whether it is supposed to be done before or after the operation. (BEFORE, AFTER) It is interesting that the trigger can be named as a table (however, it is not recommended) and that any number of triggers can be defined for the same table and event. However, it is not possible to determine the order the triggers will be called, and the trigger must not depend on each other. In triggers, recursive calling must not be used. Within trigger, it is possible perform any SQL statements – INSERT, UPDATE, DELETE. However, if you want to use SELECT, it must be used in combination with a key word INTO that ensures storing the data into the local variables.

11. Analytical tools - OLAP

11.1. OLAP - Information Under Control

Which customers bring 80% of the profit? How has the success of retaining customers changed after launching the loyalty program? How do different business events (campaigns, introduction of new products, changes in company processes, etc.) and the market (competition activities, changes in economic conditions, etc.) affect sales fluctuations? What is the result of comparison with the previous period? These and similar questions are asked by every realistic manager. Every day, they try to understand the behavior of the company based on the analysis of the data they have at disposal. In well-functioning companies, analysts are included in selected departments, who only take on this role and provide reasoned recommendations to managers. All of these people need to have prepared infrastructure and tools which would enable them to do this activity for the company.

11.2. What and how it simplifies work

The path to successful data analysis begins when they are being prepared. The first important step is to gather data from enterprise systems and arrange them into a form suitable for analysis and reporting. During this preparation, it is necessary to ensure that the various data and the quality of the data are correctly aligned. We must always document the links to the source systems so that the results are always credible and verifiable - in the case handling errors that occur in the original data. The resulting data are stored in a relational database where they are available to the extent necessary for operational analyses. Ready-to-use data are used to generate detailed reports on company performance and simpler analyses based on detailed data. However, in the following simplified example, we will require a different type of information: A nationwide company sells different products . The manager divides individual branches into regions. The category of products sold has three levels - categories, subcategories, specific product. The company has a logical definition that sales must follow a plan. The real fulfillment of the plan is then given by the sum of all products and all branches.

11.3. How to get this information?

For these needs, advanced analyses are referred to as multidimensional. The manager can monitor measurable business parameters in contexts, angles, and various detail levels. Such analysis requires OLAP (on-line analytical processing) technology to ensure that data are stored and pre-calculated in such a way that subsequent queries last for a reasonable amount of time. OLAP technology brings you the ability to work with data on

a summary level, identify the problem or an interesting area here, and proceed to the level of detail that is of interest to our decision making. A typical area where OLAP benefits are demonstrated is sales analysis.

Fig. 1: OLAP in Oracle Business Intelligence – Discoverer

Through OLAP tools, the manager can control how reality evolves against the plan. If the plan is not fulfilled, the trend curve shows this information in time, and the end of the year may not end with a fiasco. Summarized numbers can be easily broken down by a level below, then you can see a summary fulfillment of the plan in each region. An analysis can show that the problem lies in only one of them, the other regions fulfill the plan. By further drilling, they get to the branch level, identifying five of those who are significantly behind in the problematic region. The system will allow splitting to the level of categories and subcategories to the level of individual products that are not sold sucessfully at that location. Based on such information, it is possible to design and implement corrective actions to improve the situation. The example shows that OLAP data are actually stored in a tree structure - a cube. Here, for each depth level of this tree, values for the subtree are calculated. Such a hierarchical structure is definable on most enterprise data. It simplifies analysis when evaluating them. The price for this simplification is the time and place needed to calculate and save the tree. However, if the cube is filled, it can accurately respond to very complicated queries, in particular to make comparisons of time slots limited by complex conditions. A typical complicated query is, "Which ten customers saw the highest increase in turnover with my business compared to last

year, in the districts that rank among 20 % of my least profitable ones?" The result can then be used to provide extra care to these customers. With OLAP, it is possible to query data from many angles - many dimensions - and arbitrarily parameterize these angles. As we have shown, it is not appropriate to store data in relational form for these types of queries.

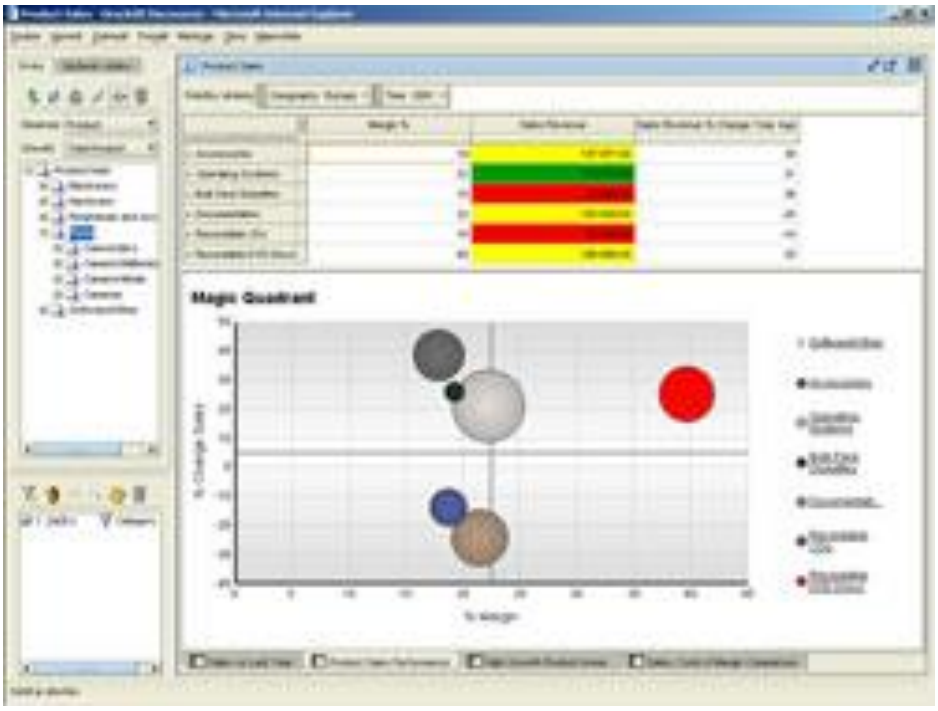


Fig. Figure 2: Sales Analysis in Oracle Business Intelligence – Discoverer

11.4. Analytical tools

Up to now, we have described ways and technologies of storing data. To work with them (performing analyses) requires appropriate tools. Choosing the right analytical tool is a key issue for effective information gathering. In particular, it is important to take into account the experience of analysts who are already working with the resources at their disposal - very often, tools from the Microsoft Office family are used. Therefore, it is important that the solution chosen allows sufficient co-operation with the tools people are used to - this leads to lower costs of adopting a more efficient way to work with information. At the same time, security of access to information, automated distribution of reports, high-quality graphic output for presentation, or integration with company intranet must also be ensured.

The most important thing are people

Despite the great number of technology, we must not forget people. One of the most valuable people the company can get is experienced analysts because only an experienced analyst can get essential information from the huge amount of data and formulate recommendations for management.

11.5. OLAP cube

OLAP cube (online analytical processing) is a method of organizing data that extends a two-dimensional table arrangement so that each data dimension is stored in one axis of the cube. This overcomes some limitations of relational databases.

Arranging data into cube vectors allows to access them from different points of view (dimensions) in the same way. This way there is no need for difficult combining of RDBMS tables, which is demanding for the system performance. s. However, physical data storage in cubes does not allow rapid editing; in such a case the entire cube needs to be reprocessed. The cube is made up of values that are categorized into dimensions. The structure is implemented by relational tables in the star or the snowflake scheme. It is typically a parent-child structure where parent elements represent consolidation of children and they themselves can be aggregated into their own parent elements.

Basic data cube operations

For analytical needs, the following data cube operations are performed to enable processing and data projection to facilitate their understanding.

Slicing a Cube: Limiting one or more dimensions to a subset of one element.

Cubing: Limiting one or more dimensions to a subset of two or more elements.

Roll up and drill down: Navigation through the data hierarchy up and down.

Pivoting: Rotating the cube in order to gain a different perspective on data relationships.

Aggregation: Consolidation according to relationships defined by formulas.

12. Specifics of database systems. Access technology to databases. Geographic information systems.

The database (or data base) is a file system with a fixed records structure. These files are interconnected using keys. In the broader sense, database includes software resources that allow you to manipulate and access the stored data. This software is called database management system (DBMS) in Czech scientific literature. Database is normally perceived - depending on the context - as both stored data and software (DBMS).

12.1. Data Access Overview in ASP.NET Technology

Visual Studio 2010

Web applications commonly access data sources for storing and retrieving dynamic data. You can write data access code by using the classes in the System. Data namespace (commonly referred to as ADO.NET) and the System.Xml namespace. This approach was common in the previous versions of ASP.NET.

However, ASP.NET technology also allows data bindings to be done declaratively. In most cases, no code is required:

- Selecting and displaying data.
- Sorting, paging and caching of data.
- Updating, inserting and deleting data.
- Data filtering using runtime parameters.
- Creating master-detail scenarios using parameters.

ASP.NET includes several types of server controls that are a part of a declarative data-binding model, including data source controls, data binding controls, and an expanding query control. These controls handle the basic tasks that are required by the stateless site model to view and update the data on the ASP.NET Web site. Controls allow you to add data binding functionality to the page without having to understand the page life cycle details.

13. Database Objects

The term "database" is often simplified to what actually a database system (database engine) or a database management system is. It does not only include tables - they are

only one of many so-called database objects (sometimes also database entities). More advanced database systems include:

- **Views**- SQL commands named and stored in the database system. You can select from them (apply the SELECT statement).
- **Indexes for each table**. Keys are defined above the individual columns of tables (one key may include more columns), and their function is to keep LUT tables at the columns over which they were defined, to avoid records duplicity or to ensure fulltext search.
- **Triggers** - a mechanism over individual rows of a table (or the table itself) that is called for after a change, deletion or addition of a line, or deletion of a row or deletion of a table and that performs a pre-programmed action (for example, checking the integrity of data, ...)
- **User-defined procedures and functions** - some database devices support storing of named pieces of code that perform a specific sequence of commands (procedures) in the database above the given tables, or return some result (user functions). They can have parameters that are mostly divided into input (IN), OUT (OUT) and I / O (INOUT).
- **Events** - procedures triggered on a specific (user-defined) date and time or repeatedly with a defined period. They can serve for maintenance, deleting temporary data, or reference integrity checking.
- **Forms** - Some database systems such as Microsoft Access allow users to create entry forms for visually friendly input. For example, the user can define the layout of each input field from a given table, labels, etc.
- **Reports** - similarly to forms, reports allow the user to define the layout with the fields of the given table, to which the current values are added. They are used for data output (printing, presentation, or plain view). Reports can be supplemented, for example, by filters that filter out only the desired records.
- **User Permissions** - better database systems offer the possibility toseparate individual levels of access to other database objects for their users. There are tens of options, with a distinction to the individual types of commands that the user will or will not have permission to use.
- **Partitioning** - a way to divide the data in a table into multiple hard disks and thereby break down the load placed on it
- **Processes** - Database tools can give an overview of the processes that are currently using their services .
- **Variable settings** - typically dozens of variables that can be reshaped and thereby affect the behavior and performance of the database tool itself.
- **Collation** - MySQL has advanced options for setting up several dozens of character sets and comparisons, collectively called collation. The collation setting can be

done on individual text columns, entire tables, and entire databases (with cascading inheritance). Collation also affects sorting, for example, the value utf8_czech_ci ensures correct sorting by Czech (ie including diacritics and including ch).

- **Visual E-R Schema** - (in MySQL INFORMATION.SCHEMA). Visual representation of relationships (s) of dependent fields (foreign keys) between tables.

13.1. Basic concepts of database technology

Introduction to database technologies

With the appearance of microcomputers and their entry into the life of each one of us, there seems to be an increasing interest in computer-processed data, not only in the sense of computing something, but above all in the sense of knowing something. The explanation is quite understandable considering that even the most sophisticated computer games are tiring over time, and that programming in Basic is a way of calculating the type of solution of a system of two equations of two unknowns.

One would like to organize and search for some of the information frequently used and changed while using it. Assuming appropriate means of storing such information in the form of data, there would be no problem to spend several evenings sitting at the computer and make, buy or exchange the relevant data himself with another similar enthusiast. An example might be a library, an office, flight ticket processing, a city museum, a hospital, a business,

The completed object of the type described can be called an information system (IS). The IS is a set of elements in the mutual information and process relationships (information processes) that process the data and ensure the communication of information between the elements. From a user's point of view, the IS should have such features so that manipulating with them (input of data, query formulation, use of application programs) is as easy as possible, and at the same time that its functions are fast and strong enough. Providing a flexible "empty" IS to a user today can be quite fast, but only with sufficient knowledge.

Our goal will be to show the so-called database processing technology. Database technology is a set of concepts, resources and techniques used to create information systems (IS). At the most basic level, we can visualize the IS architecture with the database as follows: Data is organized in a database (DB), controlled by a program package called Database Management System (DBMS). The database and the DBMS form the so-called database system (DBS). In our terminology, we can simply write DBS = DB + SBBD. IS uses data from DBS either directly or processes it with application programs.

13.2. Geographic information system

(GIS, Geographic Information System) is a geographic information system that allows you to store, manage and analyze spatial data.

Most real-world objects and phenomena occur at some place on the earth's surface (e.g. tree, house, river) or relate to a place on the earth (the citizen has a permanent residence somewhere, the product was manufactured in a particular factory). At the same time, these objects occur in the space together with many other objects and interact with each other. Therefore, knowing the location and spatial relationships between objects is very important and can play an important role in a number of human activities, from the design of the nuclear power plant to the design of the business network and the evaluation of its success.

In practice, this means that our data in the computer must include both, i.e. both the actual data on the object and its location. This data type is called geographic (or spatial) data and a computer system that allows us to store and use such data, is called the geographic information system, GIS.

Basic components of geographic information systems

Basic Components of Geographic Information Systems as Vít Voženílek in his book Geographical Information System I. The Concept, History, Basic Components describes them, are:

- Hardware
- Software
- Data
- Service staff

For the efficient operation of systems, their balance is essential. Hardware, or technical equipment, is the technical basis of geographic information systems. Software is a set of programs executing all system operations. Data is a key element of any geographic information system. GIS is a real, actual system from the organizational structure point of view. Its operation is a collection of activities that ensure the functions of the system.

13.3. Geographic data

Dimensions of geo objects

The basic division of geo objects is a division by the number of dimensions. Real objects on the Earth's surface are always three-dimensional. They are, however, transformed (transformed) into the GIS by the required level of simplification.

- 0D geo objects – non-dimensional objects, points defined only by their location. An example can be a bus stop in a GIS modeling transport or a GSM transmitter in a GIS mobile carrier modeling signal coverage.
- 1D geo objects - one-dimensional object, line sections (edges, lines), finite length and zero area. 1D geoobjects are most often modeled by roads, rivers,

- 2D geo objects - two-dimensional, polygons (polygons), with finite circumference and finishing surface.
- 3D geo objects - three-dimensional object, geometric body. In GISs, they are used exceptionally, in specific cases. The third dimension is most often modeled in the GIS by means of the so-called digital terrain model (DMT, English DEM) by the "Surface" - linked topological surfaces (2.5D).

The basis of each GIS is geographic data. Geographic data is information about the Earth's surface and the objects on it. Data can be represented as a certain layer of information that is called a theme. Each topic represents a certain element (e.g. roads, lakes, towns, etc.). Geographic data can be organized in GIS by two basic models:

13.4. Data Collection

Getting data and moving information into the system takes a lot of time. There are a number of methods used to enter GIS data stored in digital format. Exit data printed on paper or PET film can be digitized or scanned to produce digital data. The digitizer produces vector data as points, lines, and polygons. Scanning maps in the form of raster that could be further processed to produce vector data.

Collected data can directly access GIS using a technique called Coordinate Geometry. Positions from the Global Navigation Satellite System (GNSS) can also be collected and subsequently imported into GIS. The current trend in data collection that allows users to use computers in the field with the ability to edit live data by connecting to a wireless network or editing without an Internet connection. This eliminates the need to import and update office data once it has been collected by fieldwork. This includes the ability to incorporate positions obtained by laser rangefinder. New technologies enable users to create maps as well as field analyzes, making projects more efficient and mapping more accurate. Remote-sensed data also plays an important role in collecting data and consists of sensors connected to the platform. Sensors include cameras, digital scanners and LIDAR, while platforms usually consist of aircraft and satellites. In England in the mid-1990s, hybrid balloons called Helikites first promoted the use of compact digital airborne airborne cameras as Geo-Information Systems. Airplane software measuring 0.4 mm precision was used to link photographs with ground measurements. Helikites are cheap and collect more accurate data than planes. Recent development of miniature unmanned drones. For example, the Aeryon Scout dron was used to map a 50-acre plot with a sampling density of 1 inch (2.54 cm) in just 12 minutes.

13.5. Multidimensional cube

The basic building unit in the multidimensional database is cube (cube, data cube, multidimensional cube, hypercube). The cube in a multidimensional database consists of a

set of dimensions and measures.

Cube dimensions are the categories against which we want to aggregate and analyze data. Dimensions arise from relational database tables. Typical dimensions in multidimensional databases are time, position, product. Dimensions can consist of a number of levels that further refine the data.

Cube measures are the quantitative data we want to analyze. As dimensions, measures are derived from relational database tables. Typical rates are sales, expense, prices, but almost every quantitative figure can be a measure of a multidimensional dice.

13.6. Saving data in the OLAP database

The source data from the relational database can be stored using one of three basic methods:

Data cubes contain aggregated data that are the basis of multidimensional analysis. Comprehensive multidimensional applications (such as informatory) contain data related to different contexts, but some dimensions can be shared. It is therefore more understandable to create a single data cube for one context than to use a data cube embracing all contexts. The data cube consists of only those dimensional attributes that share all of its numeric attributes. This means that dimensional attributes form the basis of a data cube. If we want to get information from more data cubes, these attributes can be linked at the level of one or more common dimensions. Navigation between data cubes is not seen by the user.

Dimension tables describe specific dimension properties. For each dimensional attribute, there may be no more than one dimension table. Unlike the original star model, in our model, dimension of data cubes can be without dimension tables. On a schematic level, the dimension table consists of attribute names, and at the level of the examples it is based on the values of these attributes.

In informatory, aggregate data must often be analyzed on the basis of complex criteria that depend on the properties of the dimensions. In order to specify these criteria, the information should be used in several dimension tables. It results from this that semantic relationships between dimension tables are created based on the shared values of some attributes in dimension tables.

INFORMATION AND COMMUNICATION TECHNOLOGIES

1. Didactic means, aids, multimedia

Information and Communication Technologies, ICT, in Czech IKT, includes all information technologies used for communication and information work.

Components of didactic techniques:

- These are DT devices (devices and technical devices for the provision of visual, auditory and audio-visual information). The main feature is absolute universality.
- Teaching aids (any appropriately processed curriculum), great specificity.

1.1. Didactic aids

1. Original subjects and real facts

- natural products in the state of origin (minerals, plants, etc.), prepared (preparations, stuffings, cuttings, etc.)
- products and creations in the original state (instruments, works of art, etc.), modified (sets and sets of samples, cutting machines, etc.)
- phenomena and processes, physical, chemical, biological, social, etc.,
- sounds, real sounds, voice and musical expressions.

2. View and represent of objects and facts

- static, functional, modular, flat models, etc.,
- presentations presented directly (pictures, photographs, diagrams, etc.), presented through technical means (statically, dynamically, interactively, virtually, 3D, etc.)
- voice recordings.

3. Text aids printed or digital

- classical, working, programmed, interactive books,
- working materials, dictionaries, spreadsheets, task collections, atlases, etc.,
- additional and auxiliary literature and information sources.

4. Programmes and programmes presented (implemented) by technical means

- programmes, educational films, radio and television programmes, etc.,
- programmes, information, tutoring, repetitive, etc.

5. Special aids

- experimental sets, building kit etc.

1.2. Teaching aids

There are 2 aspects of assessing the correctness and suitability of the curriculum:

- **content point of view** - assesses whether the curriculum contained in the aid corresponds to the current level of the problem identified and is in accordance with the objective of teaching the given type of school - a matter of didactics of the given subject.
- **formal point of view** - solves the question of whether it is possible to present without difficulty to the pupils the aid given by means of DT (font size, appropriate size and form, etc.) - the subject of didactic technique.

The aid provides information:

- *Content* (related to the content submitted - concepts and relationships between them, etc.)
- *Interpretive* (shows how to work with the information provided).

The aid is very specific - it fully respects the given subject, the type of school, the year, the topic and the specific teacher.

The importance and function of teaching aids and DT means in teaching:

- Help raise the interest of pupils,
- Promote concentration of pupils,
- Promote consistency of attention,
- Motivational function,
- Lead both to learning curricula and working methods,
- Expand information opportunities in the process of teaching and learning,
- Facilitate mediation of difficult to verbally communicated subjects,
- Enable a more faithful understanding of the reality,
- Combine theory with practice.

Modernization of the teaching process:

It is not a mere introduction of modern DT resources into the teaching process, but the next sequence of phases.

- Modernization of content of teaching process,
- Modernization of teaching methods,
- Inclusion of modern DT means.

1.3. Multimedia

Are the area of information and communication technologies, which is characterized by the merging of audio-visual technical devices with computers or other devices.

A multimedia system means a collection of technical resources (such as a personal computer, sound card, video card or video card, camera, CD-ROM or DVD drive, relevant service software, etc.) that is suitable for an interactive audio-visual presentation. Since the early 1990s, the use of multimedia apps or multimedia software has been used to combine text, image, sound, animation, or movie data.

2. Didactic technique

Didactic equipment (DT): devices and technical devices for the provision of visual, auditory and audio-visual information.

- Prerequisites for the use of teaching means / DT /:
- Thorough analysis of the lesson: determining the structure and center of gravity of the lesson.
- We select knowledge to clarify the learning aids necessary.
- We will derive the technical form of the aid and its inclusion in the lesson from its function in the process of learning the given knowledge.
- Evaluation of the relationship between the teaching aid used during the lesson and the materials available to pupils for their own home preparation (in textbooks, notebooks, etc.) Is it sufficient to use the aid during the class or is it appropriate for pupils to be available for homework preparation?

2.1. Multimedia didactic means

Multimedia - systems for technical elements for interactive audio-visual presentation; Allow the user to work with several types of media (media): text, image, sound, computer graphics and animation, video, and interactivity between the system and its user. I.e. Multimedia is characterized by the following features:

By text ? Audio-visual ? Image ? Animation ? Video ? Interactivity ?

Interactivity may include various options:

- the user selects content, combines its parts,
- the user influences the pace at which the multimedia system presents information,
- the user exchanges information with the system (e. g. by question-answer)
- the user controls the process by which the system presents information,
- the user inserts the content of the system, compares it, etc.

Multimedia didactic means – e. g.:

- educational software (e. g. interactive multimedia DUMs), multimedia presentations, multimedia recordings of teaching - e. g. video lectures, didactic computer games,
- Learning Management System (LMS): planning, creating, presenting and managing content as well as suitable environment, with tools for both LEKTOR and CLIENT and their mutual communication (e. g. Moodle).

Distribution of didactic techniques (depending on which human senses we use when using DT):

- Visual aids - it affects Vision.
- Auditory Techniques - it affects Hearing,
- Audio-visual Techniques – it affects Vision and Hearing at the same time,

Hypermedia and hypertext didactic means:

- Hypertext - a text composed of blocks of words or symbols electronically linked by paths (electronic lines) in an open and still unfinished texture structure (network).
- Hypermedia - A digital resource that includes active links not only to texts, but also spreadsheets, animations, pictures, audio, video.
- Hypertext and hypermedia didactic resources - Digital resources containing hypertext and hypermedia elements. They are in the form of a network in which the main lines of the text are interconnected
- Text lines and media elements that develop, complement, illustrate, the main text. The learning subject proceeds in these networks in a unique, individual, unpredictable way.

3. Modelling. Definition. Theoretical models.

The term "model" is used in different contexts. This term means:

- Mathematical and mental approaches to the problem,
- Didactic means.

In the spoken language, the term most often means:

- The original pattern of something created;
- A prototype, such as a car model;
- The ideal pattern we want to implement in a particular activity or function, such as the model of a perfect teacher;
- The structure of what is in our interest, such as the model of the school;
- In art, it is a thing or a person to be painted, carved.

The ideal model is a display of the phenomenon under investigation, which may contain hypothetical explanations and can help verify the correctness of the hypothesis. Ideal models that are associated with a particular theory are called theoretical models. Material models are existing objects whose properties permit the reconstruction of the structure or the essence of the subject being studied or the course of the process.

These are simplified and distorted reproductions of ideal models and can therefore be recognized as "model of models". Before the material model arises, it must exist in the scientist's mind as a certain idea, that is, as an ideal model. It follows that models are in their original form abstract concepts that can later be rendered in a concrete way. For scientific and educational purposes, the most important are those models that are linked to the creation of new knowledge.

Theoretical models

They can perform the following functions:

- To pass on information to the client about the theory and its interconnection with relevant experimental data;
- To familiarize clients with concepts in the field of the theory and to develop the ability to use models to solve given problems;
- Verify the model with experimental confirmation or denial of assumptions based on it in order to familiarize clients with the scope and conditions of its use;
- Verbalize the model, which leads to the formulation of the assumptions of the relevant theory.

Models can be divided into static and dynamic:

- **Static models** - mostly folding and made in cut, perfect image and faster understanding of the function.
- **Dynamic models** - they are models that imitate movement and function.

Simulators - these are models of real objects, devices, etc., which can be prepared and perfected for the use of real devices.

Virtual computer models - these are models that simulate some phenomena, objects, objects in three-dimensional or two-dimensional space.

Function of models, technical teaching resources:

- Basic functions: information, formative, instrumental.
- Didactic function: motivational and stimulating function, rationalization in relation to teacher and pupils, reinforcement of information repetition, systemization (incorporation of information in the system of previously acquired knowledge), control and control function.
- Ergonomic function: (the doctrine of relations between man and the working environment and the means of work that are most suitable for the working environment) managing: e. g. reducing unnecessary time for teachers and pupils, full use for teaching management, regulating your own learning pace according to dispositions and the state of the psyche.

ACHIEVEMENTS OF A NEW, DYNAMIC COMPUTER MODEL

The first and most important assumption is that the newly designed microscope model and its corresponding visual teaching aids are basically correct.

4. Appearance of visualization

To view objects in the learning process, more interactive 3D models are currently being used, offering different viewing possibilities in varying degrees of detail, as opposed to 2D visualization, which is not sufficiently clear and does not fully reflect reality, and sometimes it happens that some processes modelled in the background, inadvertently merge together). 3D interactive models bring us closer to the real reality that we can see or examine in the real world with difficulties or it does no longer exist. The information obtained can then be associated with the "experienced" experience, remembering them better and "re-equipping" if necessary. In our paintings, we and our experience mirror us.

In the picture, it constantly creates an attractive force and mystery at the emotional level, which stimulates man to interactivity, social communication, and to create their own knowledge, knowledge structures, and critical assessment of information (Scruton, 2005).

Means of visual technique

prostředek DT	pomůcka (nosič informace)
tabule	pedagogická kresba
magnetická tabule	papírové materiály, předměty na feritech
flanelová tabule	pomůcky podlepené flanelem nebo na suchém zipu
plexitová tabule	pedagogická kresba
flipchart	pedagogická kresba
stojan nebo háček na tabuli	nástěnná tabule, obraz nebo mapa

- for non-screened aids

Means of DTAid (information carrier) board Pedagogical drawing Magnetic board Paper materials

Objects on ferrite Flannel board Tools taped with flannel or on Velcro fastener Plexit board Pedagogical drawing flipchart Pedagogical drawing Stand or hook on the board Wallboard, picture or map

- for static projection

prostředek DT	pomůcka (nosič informace)
epiprojektor	pomůcka na neprůhledné podložce
diaprojektor	diapozitiv, diapás
zpětný projektor	pomůcka na průhledné podložce nebo velkoplošný diapozitiv

Means of DTAid (information carrier) EpiprojectorA tool on an opaque pad projector Slide, diabelt Overhead projector Aid on a transparent surface or a large slide

- for dynamic projection

prostředek DT	pomůcka (nosič informace)
filmový projektor	němý film, filmová smyčka

Means of DTAid (information carrier) Film projector Silent movie, film loop

Constructivism and visualization

In recent years, implementation of constructivism in both technical and multimedia, humanitarian and foreign language education.

The constructivist concept is referred to as the ideal pedagogical basis for visualization, ie for the visualization of reality perceived through visual receptors

Constructivist pedagogy puts the client at the centre of the learning process. Similarly, the visualization associated with the application of the principle of clarity assumes an independent client who can partly manage and organize his learning.

The traditional role of a teacher naturally changes – the teacher becomes a constructivist tutor, facilitator and guide.

Communication between actors should be encouraged and activated both by creating a pleasant atmosphere of open space for sharing opinions and by a suitable concept of group / cooperative or collaborative / work.

The main features of teaching can be considered the transition from transmission teaching, so-called "you -education" to self-evaluation, i.e. self-organization, self-determination, self-education.

Media approaches

- Media Optimism - The optimistic reception of the media, considering the media to be a pervasive businessman. Transhumanism, Extropism, Singularitarianism,

Techno-utopism.

- Media Pessimism - Critical opponents of media-optimistic directions pointing to the negative aspects of technological and media development, rejecting the merger of man and technology (media).
- Mediacism - too much reliance on the media. Belief in the fact that mankind will be able to fully control the technological media and all the problems it involves and to use it effectively for its well-being.

5. The role of image in culture. Teaching using an image.

THE PICTURE AND THE DRAWNING ACCOMPANY PEOPLE FROM THE VERY BEGINNING TIME. The oldest paintings we know today are paintings in the Chauvet Cave in France (31,000 ± 1300 BC). From this moment on, the person constantly accompanies an image that over time has various informative and aesthetic functions. The important educational role of visualization can be noticed in the Middle Ages. Typical example of the then "picture education" can be the Bible Pauperum (Bible of the Poor).

It was very common to link the picture to the text in order to more fully illustrate a particular story. Over the course of a century, symbols, allegories and emblems that represented the text appeared in many paintings; on the contrary, the text itself was often accompanied by images, often in the form of initials or illuminations.

John Amos Comenius, who wrote Orbis sensualium pictus in 1658, also recorded the tradition of teaching with the aid of painting. It was an illustrated "Encyclopaedia", divided into 150 chapters with 150 woodcuts. The world in the pictures showed and named all the things of the spiritual and material world that were necessary for the children.

Visualization

Images can not only serve the role of illustration, but sometimes it is also possible to explain problems that are hard to imagine. As an example of such an approach was the text of prof. Janusz Rachonia of Gdańsk Polytechnic, which describes how he explains the resonance theory of Linus Pauling to the students by painting by Salvador Dalí.

6. Media communication. Social communication: verbal, non-verbal communication.

Media Education

The relationship between media education and media literacy is simply a relationship between the means and the goal. Media education is thus defined by means of media literacy as education for orientation in mass media, their use and at the same time their critical assessment as a deliberate educational influence on achievement of a certain degree of media literacy, or simply as education for life with the media. Media education was gradually formed, and its roots are sometimes laid at Comenius, sometimes up to ancient Greece, but the real development came after the Second World War. Media communication has tremendous power, it creates its reality; Used to increase the range or self-presentation; It is good to maintain a positive relationship with the media; Neighbouring sectors - public relations and media relations.

Types of communication media

- Primary - "sets of characters and rules for their use (native language)"
- Secondary - means of recording and transmission of messages (pictures, fonts, print, transmission and broadcast technology, computer communication networks).

The word communication comes from the Latin ***communicare***, which means "to share something together, to do something in common. Communication is a basic condition for the existence of every social relationship. It is also a means of socially integrating the individual into the human community. For a perfect interaction with the environment, it is necessary to learn to listen to the inner impulse - to consciously observe your thoughts, to cultivate a constant inner conversation, to monitor feelings and to learn to understand them.

Communication can be divided into:

- **Intrapersonal communication** takes place inside an individual and takes the form of an internal dialogue. It is a "self-talk", a self-reflection of one's own behaviour and communication with the outside.
- **Interpersonal communication** takes place between two or more people between whom there is a relationship. A specific type of interpersonal communication is group communication.
- **Mass communication is** characterized by a one-way flow of information from one and more communicators (resources) to many communicators (recipients). This is communication through media, such as radio, television, the press, and the Internet. There is no direct feedback in mass communication, so it is important to

critically evaluate the information received.

Vybíral (2009) defines five basic communication functions: inform, instruct, convince, negotiate, entertain. Social communication is a condition and prerequisite for the existence of any human community.

Communication factors:

- Source of communication- communicator,
- Determination of communication / recipient /
- Communicant,
- Communication / communiqué /,
- Communication space - channel.

7. Media education as defence against the negative effects of media communications.

The relationship between media education and media literacy is simply a relationship between the means and the goal. Media education is thus defined by means of media literacy as education for orientation in mass media, their use and at the same time their critical assessment as a deliberate educational influence on achievement of a certain degree of media literacy, or simply as education for life with the media. Media education was gradually formed, and its roots are sometimes laid at Comenius, sometimes up to ancient Greece, but the real development came after the Second World War.

Reasons for Media Education

- Lots of information, knowledge, opinions;
- The changing conditions of human life;
- Development of media literacy as part of general education;

Media literacy = a set of competencies that help the user to search, analyse and evaluate information, and pass it on;

The content and the way of implementation are different for different education systems.

Conceptual questions:

- What should be the content of media education?
- What is the mission of media education?
- What is the way of realizing media education?

Concept of media education

Two basic components:

- knowledge - typical of Canada and Scandinavia - is based on a critical branch of media theories, the Frankfurt School, and British cultural studies.
- skill - typical of the USA, assumes that pupils acquire the necessary knowledge and skills by trying out how the media work.

Negative effects of media communications

Radio

I believe that the role of radio as a means of influencing the views and the value system of children and youth is at present the smallest of all information technologies. Most young people use radio only as a sound background. The only thing that can affect the listener is the necessary ads, which are often very subdued by the melodious melodies and the audience's memory, and some slogans are constantly remembered.

TV

One of the most important media on the child's psyche is television. When a child watches 24 McLUHAN TV, Herbert for an action movie with bad content, he considers everything that goes with it for granted and he also wants to try it out without even thinking about the consequences.

DVD player

Children spend their time looking at DVD discs instead of going out with friends, but this hobby can also lead to copying DVDs that are illegal from the point of view of copyright. Children watching a movie at DVD do not develop their rhetorical skills, they do not connect fantasy. Which can lead to small vocabulary, poor speech and obesity.

Mobile phone

With a mobile phone, there is a danger that school students will use their phone for cheating. It leads to inattention during an hour, to a lack of concentration. Often used to play games, to write SMS while teaching.

An example of negative effects

- Everyday communication takes place via e-mail,
- We manage finances with e-banking,
- We read news mostly on the Internet,
- The influence of the media on the individual's psyche - especially on children and youth,
- Social networks - children,
- The most effective way to alleviate the negative effects is the family.

8. Photos, its history. Digital Present and Future.

The history of photography

Even the ancient Greeks have found that when the light passes through a small hole in the dark room, it shows the image of what is out there. In the 9th century BC this invention was used by Arabs in astronomy to determine the location of the Sun or solar eclipses. Leonardo da Vinci, an Italian scholar of the 15th century, described the phenomenon in detail as a "camera obscura", or a dark room.

In the 16th century, the Italian Giambattista Della Porta used a lens instead of a small hole to give a sharper image. Since then, camera obscura has appeared in all sizes, ranging from the smallest sizes to the large ones on the lookout towers and beacons.

In the 16th and 17th centuries chemists came to the point that some substances, if left in open space, changed their color. 1725 - Johann Heinrich Schulz discovered that silver salts are sensitive to light. However, it took almost a hundred years before the combination of these two separate discoveries created the first image on paper.

With the invention of photography, a new kind of image arose, in which the technique and mechanics taking over the manual work in creating a record of reality played a significant role in its invention. The new technology was no longer based on the handmade but the photochemical process.

Digital photo: The emergence of digital camera technology is associated with television image recording technology. In 1951, for the first time, tape recorder (VTR) recorded the image from the television camera by conversion to electrical impulses and deposited it on a magnetic tape in Bing Crosby's laboratories (a research team sponsored by Crosby led by John Mullin). In 1956, RTD technology improved through the discovery of Charles P. Ginsburg (Ampex Corp.) and came into normal use in the television industry. TV and video cameras use a very similar technology to CCD scanners for digital cameras.

In the mid-1970s, Kodak discovered several sensors capable of capturing static images that also worked on the principle of light transmission. In 1986, Kodak researchers discovered the world's first megapixel (MPix) sensor capable of recording 1.4 million pixels (which means recording a photo of 10x15 cm in photo quality). In 1990 Kodak developed the Photo CD system - the world's first standard for colour definition in the digital environment of computers (and peripherals - printers). Especially in the years 2000 and 2001, digital photos have grown up on the technological side. From 0.3 MPix sensors to commercially available professional CMOS sensors with 22 MPix resolution. From plastic lenses to high-end ultraviolet lenses.

Important personalities in photography

Joseph Niecephore Niepce

Luis Jacques Mandel Daguerre

9. Film, its history, development and future. Film technology.

The film is a composite sign for cinematography and all its aspects - artistic, technical, commercial and social.

Way of passing information, source of entertainment, means of communication.

While watching the movie, the human eye sees a sequence of consecutive images. Rapid rotation of the projected images in a very short period of time with our eye nerve and consequently the brain appears as an uninterrupted activity (the image is not torn and is perceived by the brain as a continuous motion).

History

- 1893 - kinetoscope - William Kennedy Laurie Dickson.
- Viewer for one spectator.
- The film could be watched only through a small peephole on the body of the device.

The development of film as a technical invention as a kind of art, cultural, social and economic phenomenon occurred in the late 19th century. The history of the film as an independent artistic form begins in 1895, when a cinematographer of the Lumiers Brothers (28th December 1895 in Paris) was presented with great acclaim in Paris. The film was 35 mm wide and 16 frames per second (fps). The Lumiers Brothers dealt with documentary and travel videos. The first short feature film is considered to be their Upper Spider. Comedy with a gardener and irrigation hose

The film originated as a technical miracle, the youngest, the most synthetic and audiovisual art, whose main attribute is iconicity. The film's inventor is T.A. Edison, but the movie's motives can be found in so-called animated photographs. There are two basic paradigms: passive capture of reality (documentary line - Lumiers) and science fiction, trick line (Mélies). Georges Mélies - the first film studio in Europe in 1897. Author of film tricks.

In the 20s of the 20th century there was a change that completely changed the form of the film. In California, there was a time to set up film studios. The quality of the films played an increasingly important role in the competitive struggle, and it was necessary to gain the audience with something that differed from the average production.

Various film sizes and film editing allowed the film tension to grow and lead to narratives on several levels, for example difficult to realize in the theatre. Deepening and refining the film expression in the area of editing and using the camera. With the use of multiple lenses, the range of shots ranging from large units to detail has been fully utilized, and the camera has also been detached from the place and used to ride. During this period, the film was completely detached from the theatre, it found its visual rhyth-

mic way of expression, but also a new cinematic way of acting, writing scenarios,

Silent movie

Charlie Chaplin - one of the most famous and best-paid artists. Every time, he played socially declassified characters that fight against the beautiful and successful people.

Joseph "Buster" Keaton, created a new, completely distinctive type of film comedy. The character of his film character was a stark, stone expression of his face, which became his "corporate brand" and for which he earned the nickname "The Great Stone Face".

Sound movie

In the early 1990s, Edison and Dickson designed a prototype of a sound film, a system called Kinetofonograph or Kinetofon - an ancestor of the kinescope giving non-synchronous sound. The first known (and also the only surviving) movie with live-recorded sound was the kinetophone test - Dickson's 17th overall soundtrack.

The Jazz Singer (1927), launched by Warner Bros., began the era of soundtrack. In the next five years, all technology, instrumentation and the face of the film have changed fundamentally. The sound made the audience more interested in the film. However, many actors who did not speak English lost their jobs, as well as pianists who made soundtracks in silent cinemas.

Soviet soundtrack appeared a little later. This delay was caused by the production of own cinematographic material, which did not allow the Soviets to pay fees for foreign material. Still in 1930, silent films were filmed, such as the Ball. The first English soundtrack was recorded by Hitchcock in 1929.

The 1930s were a great Hollywood era, when many new genres emerged and gradually became a "dream factory" whose thrilling or merry films allowed people a short-term escape from the depressing reality of their everyday life. Almost all the genres reproduced the myth of the land of limitless possibilities and swore in infinite variations to the "American dream" of a great career "from a dishwasher to a millionaire.

Film production in Czechoslovakia

By the end of 1941, all Czech production companies were destroyed and only two production companies - Lucernafilm and Nationalfilm - were left behind. All the rentals were unified into the Kosmos monopoly, which had the only right to distribute Czech films beside Lucernafilm and Nationalfilm.

10. Audio media, their history, present and future.

Auditory aids

According to Janíková (2005), auditory aids play an important role in "mediating and practicing the phonological aspect of the lexical unit." (Janíková 2005, p. 129) They also help pupils at the beginning of the process of learning a foreign language and when they are familiar with the new vocabulary. When using authentic material, the pupil can listen to the pronunciation of native speakers using auditory aids. You can use recordings on a tape recorder or CD, news or radio or television advertising, and more. For many pupils, listening to an authentic recording medium, such as a native speaker, provides auditory media.

Audio aids

prostředek DT	pomůcka (nosič informace)
gramofon	gramofonová deska
kotoučový magnetofon	magnetofonový pásek
kazetový magnetofon	magnetofonová kazeta
přehrávač CD	CD (kompaktní disk)
rozhlasový přijímač	rozhlasové vysílání
audioknihy	mp3 záznam

Means of DTAid (information carrier): gramophone, gramophone record, disc, tape recorder, tape, cassette recorder, Tape recorder, CD player, CD disc, wireless set, radio broadcasting, audiobooks, MP3

Audio-visual equipment

prostředek DT	pomůcka (nosič informace)
filmový projektor	zvukový film
televizor	televizní vysílání
videomagnetofon	videokazeta
multimediální počítače	CD-Romy s multimediálními programy
dataprojektory	data v počítačích či jiných zdrojích

DVD přehrávač	DVD

Means of DTAid (information carrier): Film projector, sound movie, TV, TV broadcasting, video recorder, video tape, multimedia computers, CD ROMs with multimedia programmes, data projector, data in computers and other sources, DVD player, DVD

Audio Media

Auditive = hearing; Media = medium, between, in the middle. The media is an intermediary, it mediates something.

Definition - it is difficult to define the term; there are many definitions. The role of media: communication, transport of information. The term "medium" refers to technical means and the system of social institutions serving communications. Another possible definition of this concept is that "the media are social institutions that play a large part in ensuring communication in the public sphere, thus contributing to the development, establishment and transformation of culture, that is to say shared values, values and interpretations of the world" (JIRÁK, J., KÖPPLOVÁ, B., 2003. 208, p.52)

What are the audio-visual media?

This term has two meanings. We can label media with audio-visual content - that is, (Such as DVDs, video cassettes), or it is a communication medium that acts on these senses. 4 Milan Šmíd states that this is a technical term in the division of media that is used in France, where audio-visual media also includes radio, which can be a bit problematic as the radio only transmits auditory information.

On the other hand, Anglo-Saxon literature divides the media into printed and electronic. Electronic media include radio, television, audio-visual and auditory records. American literature, in this division, includes, among electronic media, a film that had only few common features in the past with electronics. At present, however, it is already connected with electronic media by close links.

Present

Auditive media - still used, often linked to visual or audio-visual media.

Players - phones, tablets, computers, television, mp3 / 4, players, radio, radio, dictaphones, man :-)

Audiobook = sound record of the spoken word; Has already begun in history, now boom thanks to smart phones, internet downloading.

Future

Do we expect some future for auditory media? Will a TV screen or a radio be part of our home, as it is so far? Will the TVs in our homes be replaced by projectors?

There is a new concept: projectors that will need little space for display - a huge image at a distance of 25cm, complicated installation, connection to set-top-box, computer or

game console with wireless system; Projectors for mobile phones and tablets / Lenovo, ... /. Reducing the price of projectors will allow replacement of TVs.

11. TV World in History. Analog and digital.

History of television in Czechoslovakia

Pioneers:

František Pilát - the later post-war technical director of the Barrandov Film Studio, himself built a television set. Pilate was the first in Czechoslovakia to receive experimental Baird's "thirty-line" broadcast, spread out in the early 1930s (1929-1935) from Great Britain to a medium wave of 261.5 meters.

The most active pre-war pioneer of the television is dr. **Jaroslav Šafránek**, associate professor of experimental physics at Charles University in Prague. In 1935 Šafránek built his own functioning television equipment, with which he later traveled to the Republic and publicly presented it. The Ministry of Post and Telegraph refused to authorize Šafránek to broadcast television in the air. Šafránek´s equipment could only work in the laboratory and lecture halls. While Šafránek, radio amateurs and their interest organization, Czechoslovak Radio Broadcasting Corporation requested permission for experimental broadcasting of a mechanical low-line (30 line) television mainly serving radio amateurs, the Ministry of Post and Telegraph, which since 1934 closely watched developments abroad, wanted to provide frequencies for television broadcasting to some more developed Projects. It was guided by the principle - to wait, to study foreign facts and then to decide.

In 1939 television research was completed on the territory of the former Czechoslovakia. (Threats to the Republic, Munich, and the Nazi occupation). At that time, Šafránek was working on a more advanced 240-line image decomposition device.

On November 17, Germans concluded Czech universities.

Šafránek's television experiments ended. He lost his position at the university, and his German Institute of Physics was closed by the German authorities. Šafránek allegedly managed to take some equipment to the Pardubice Telegraph factory, where he stayed throughout the war.

Even before the end of the war, in April 1945, the top German experts left Fernseh A. G.. They move to Austria. In May, the factory is hired by Czech local authorities, the factory in Smržovka Fernseh A. G. immediately renamed to Televid. At the beginning of June, Televid is took over by the Czechoslovak Ministry of Defense under its administration.

However, in July 1945, the security of the company was taken over by the Soviet military administration, for which the factory was part of the war booty, and several Soviet experts came from the Leningrad Television Institute. At this time Associate Professor Šafránek often comes from Prague to Smrzovka. But according to memories, witnesses did not intervene in technical development, because his 240-line mechanical system was outdated and Televid worked on an electronic, later "European" standard of 625 lines. However, Šafránek's name has once again been written into the history of our television. He organized the internship of a group of 25 experts who joined Smržovka - after the

agreement of the Czech authorities with the Soviet military administration - in October 1945. Before the trainees could work actively, the Soviet party decided to move Televid as a war booty.

Jaroslav Šafránek is attributed to the primacy of popularization of television in Czechoslovakia. He published the book Televise, in which he acquainted himself with the technical principles of image transmission at a distance. An up-to-date version called "Televise - The Physical and Technical Foundations of the Television" by Šafránek was published after the war. Šafránek tried to distinguish the simple technology of transfering the moving picture from the complex television broadcasting process, for the television as a mass medium, where he coined the word "roar", which, according to him, "correctly describes the essence of television.

On March 23, 1948, journalists were invited to Tanvald, welcomed here by VTÚ General Josef Trejbal and Technical Deputy of Czech Radio - Kazimír Stahl. As part of the demonstrated technique, even with the use of some trophy components, the design itself was a television receiver with a 16x21 cm own production screen.

There were two television cameras at the MEVRO exhibition in Prague and the signal was transferred to the receivers by cable. A month after the end of the MEVRO exhibition, July 4, 1948, during transfers from XI. Sokol organization meeting, three cameras were working on the Strahov stadium, and the signal was transmitted by air from the mast of Petřín for 25 receivers in various institutions and in public places (e. g. Exhibition Grounds, ČSS Radio, Red Law paper Redaction, reception was also checked outside of Prague in South Bohemia and the Giant Mountains).

From 1949 to 1952, television in Czechoslovakia ceased to exist.

The television equipment of the VTU Military Technical Institute, including two camera chains and ten television sets, was transferred to Czechoslovak Radio, which was established at the beginning of 1949 by the ÚRT Institute of Radio Technology. Although he continues to carry out the task of preparing television broadcasts during the first five-year period, ie by the end of 1953, but the Cold War, which was intensified in 1950 by the conflict in Korea, caused the ÚRT Office had no other office to cooperate with, technical research has focused exclusively on military needs. According to Ing. František Křížka In 1951, the Office of the Czech Radio organized several experimental broadcasts in its building in Vokovice and lent the receivers to party and government officials to promote television without success. Turnover occurs in 1952.

On April 8, 1952, the Government issued a regulation requiring the Ministry of Communications and Welfare "to build and operate technical radio and television equipment".

The first "Programme Director and Director of Television Studies", established within Czechoslovak Radio Karel Kohout, was appointed on 1 February 1953 (three months before the scheduled start of the broadcast). Karel Kohout came from Barrandov's studio and started with his legendary secretary Maria Kořenová to organize a broadcast from a temporary office on Wenceslas Square.

For several years the broadcast has been limited to the Petřín transmitter in Prague

reaching the Central Bohemian region to the foothills of the Jizera Mountains and Krkonoše Mountains. At the end of 1953, some 2000 TVs were in operation, of which a thousand Leningrad brands were imported from the GDR, where they were then produced in the Soviet license. But as early as 1953, Tesla supplied the Tesla 4001A to the market. They were selling for CZK 4,000 (at that time it was an almost half-year average salary). At the beginning of January 1955, when the so-called concessionary fee began to apply, the statistics of 3833 concessionaires were reported. By the late 1950s, the television receiver became a scarce product on the market.

On February 11, 1955, the first direct TV transmission of a sports match in the history of Czechoslovak Television took place.

On April 17, 1955, the first direct transmission of the opera from the National Theater took place.

Since October 1955 it has been broadcast 6 times a week (not on Mondays); From December 29, 1958, began broadcast nationwide every day, seven days a week.

On New Year's Eve on December 31, 1955, the second Czechoslovak television broadcaster Ostrava-Hošťálkovice started operation.

TV studio in Brno was established only in 1961; On February 25, 1962, TV broadcasting began in Košice.

This basic structure of the five major television studios resisted in fact until the breakup of Czechoslovakia in 1993. At the turn of the 1950s and 1960s, a network of transmitters and converters was built so that in 1961 the television signal covered all regional centers and most of Czechoslovakia , so that shortly thereafter (1962) there were more than one million television owners.

On 1 January 1993, after the dissolution of the Federation, Czech Television was established. In the 1990s, the first private TV company was launched (NOVA and others). In 2000, the Czech Republic was preparing for the transition to digital TV broadcasting in the digital terrestrial (formerly termed Terrestrial) DVB-T platform, which was supposed to replace analogue terrestrial broadcasting. Technically, the Czech Republic has been very well prepared - the experimental digital television broadcasts have already been successful in the three largest cities.

The difference between analogue and digital transmissions can be likened to sending money by a coach or by bank transfer. If we send the money to a shoemaker, the recipient will receive our money physically as we sent it, provided that the coach does not declare that the courier is not lost anywhere, etc. While we send them by bank transfer, the recipient receives physically other money, but in any case at the same value as they were sent.

Digital transmission of information is the transmission of a value, in the form of a number. (In this case the so-called binary number).

An analogue broadcast is an outdated way of spreading television and radio signals. Both the image and the sound are transmitted by electromagnetic waves. Color and

sound information is generated by modulation of this continuous (analogue) signal. Each TV or radio frequency thus bears one station signal. Analog broadcasting is currently replaced by digital broadcasting. In the Czech Republic, analogue broadcasting was turned off at the end of 2011, when analogue darkness occurred.

Digital Broadcasting (DVB = Digital Video Broadcasting) allows the multiplex to transmit several TV programmes at one frequency. This makes it easier to use the bandwidth used for analogue TV broadcasting.

12. Multimedia in the mirror of time. Philosophy of the media.

What are the media? Media is often referred to as a means of communication, a media, most often a technical device for communication between the communicator and the recipient. In addition, they are mass media, so-called promoters. Other media than mass media play an important role in the promotion. And, as examples, we can present fairs, exhibitions, showcases, covers, but also lectures, excursions.

Newspapers, magazines, films, radio and television broadcasts, the Internet (news, educational, entertainment portals, social networking, blogs, spoken word, films, music) - communication resources available to a potentially large number of users at regular intervals.

Through them, we can purposefully influence public opinion, manipulation, deliberate message selection and shrinking, mixing of messages and evaluating comments may occure, exaggeration (positive, news sound more positive, negative news sound more negative).

The force of editing (only a part of the sentence is left), manipulation with the image (the order of the images may cause confusion of cause and effect). Speech rendering.

Media features

- INFORMATION FUNCTIONS. It is mainly news and journalism, where news, comments and reports are used.
- FUNCTIONAL FUNCTION. Mass media bring music, art, sports, and even today's phenomena - television shows, reality shows, cooking.
- KOMERČNÍ FUNKCE provides the media with funding for broadcasting and guarantees profit, it is realized mainly by advertising, but also by teleshopping and sponsoring.
- EDUCATION

Political, social scope.

From 1900 to 1925, the journalistic industry, the first field reports and photojournalism, emerged.

1918 - Czechoslovak Press Office "CTK" is founded; Political diaries, independent diaries, btabloits. Although the system of the First Republic is today a model of the democratic system for many, the functioning of the periodical press has been much more limited at this time than today.

Role of radio - entertainment, news

In 1925, news began to play a new and more active role on the radio, not just reporting messages, but trying to bring events closer. First attempts at reporting. The first sports report.

Radio and education

Gradually, radio courses for foreign languages such as French were also included in the programme offer. Lectures, interviews, discussions, bands, radio games, professional broadcasting.

1930-1960

TV

A pioneer in our country was Jaroslav Šafránek (associate professor of experimental physics at Charles University in Prague) in the 1930s. In 1935 he completed the first television reception apparatus in Czechoslovakia. Great interest in television was shown by Czechoslovak radio amateurs.

The force of interview continues

1939 - Radio broadcasting to the Protectorate (all employees of Jewish descent have to leave).

Film and the varnish popaganda

There was a series of films depicting the territorial expansion of the Third Reich. These films did not have any great artistic value.

Media and communism

In 1948 CTK was under Communist dictatorship. It serves as the instrument of political propaganda of the ruling party. There is strong censorship, the nationalization of the radio.

Two types of reporting: for the public and non-public (for high party and state officials). The agency is formally subordinated to the government, but in fact it is governed by the Central Committee of the Communist Party of Czechoslovakia.

Years of the 50th - Sharpened Ideological War

For the next 40 years, the media in Czechoslovakia has begun to serve "the people and the Communist Party".

1952 - Establishment of the Main Press Office Administration.

1952 - Czechoslovak Radio began to cancel the broadcasts of Free Europe (broadcasting started in 1950)

Media before the occupation of 1968

Radio: natural civil speech, criticism, openness, reception of world radio stations - expansion of supply but weakening of state control.

Television: only one programme, a huge increase of concessionaires, "Sugar" - completion of the construction of a basic network of transmitters.

1968 Occupation - Time of Normalization

ROZHLAS: jammers of foreign stations (Free Europe, Voice of America).

TELEVISION: Transmitter for the Second Programme, 1973 - First color broadcast, personnel cleansing, renewal of censorship, ideological conception, attacks on dissent.

PRINT: Illegal publishing of newspapers and magazines.

80s and others

1986 - Chernobyl.

1989 - The Velvet Revolution.

A big change is the development of personal computers

1981 - 4th generation computers.

1990-2017

Beginning of the 90s - media transformation, end of censorship - the media are heading for personal ownership, the media are an institution of freedom of speech, a forum for discussion of public interest issues and private business.

1991 - Introduced 1st Web Browser - WorldWideWeb at CERN.

INTRODUCTION TO JAVA PROGRAMMING

1. Introduction to programming in JAVA

Creating programs for Java Wageform is done in two phases. In the first step we write the so-called source code. The source code is a program notation. In the second step, the source code is translated into byte code. Translation of the source code into the byte code is provided by the compiler. Bytecode are Java Virtual Machine (JVM) instructions. Each instruction consists of an opcode and no or several operands. JVM allows you to run the byte code. It functions as an interpreter, ie it reads instructions and executes them immediately. Modern JVMs often include a Just-In-Time compiler (JIT compiler) that translates JVM instructions into the native processor code. Native code consists of instructions that are specific to the processor type. E.g. Intel and SPARC processors have different sets of instructions. Because a program in native code runs faster than byte code, translating it into native code can significantly speed up the program. However, the time needed for the translation of the byte code into the native code must be included in the total program runtime. This time is not negligible, therefore, only parts that are performed repeatedly are usually translated.

1.1. Classes

The Java programme consists of one or more classes. We declare the class using the keyword class. A class can contain methods. Each class and method has a name (except for anonymous classes, which will not be discussed here). Both the class and method contents are enclosed in braces {and}. We write commands into the methods. Each program must contain the main method, which always has the same header:

```
public static void main( String[] args )
```

The meaning of each word in this declaration will be discussed later. To begin with, we will write commands only to the main method. This method serves as an entry point to the application. To start the program, JVM calls this method. The commands are executed one by one in the order in which they are written. Use the System.out.println () method to exit the screen. A program that prints a simple greeting looks like this:

```
public class Prvni {
   public static void main( String[] args ) {
      System.out.println( "ahoj" );
   }
```

```
}
```

This source declares the First class that contains the main method

1.2. Java virtual machine

Java Virtual Machine (JVM). JVM allows you to run Java programs. JVM's task is to provide programs with the same environment. The same environment ensures that the Java program runs on a SPARC-based computer running Solaris as well as an Intel-based computer running MS Windows. Starting the Java program is done in two steps: in the first step we start the JVM and in the second step the JVM loads the byte code into memory and runs it (it calls the main method). Program memory is divided into three areas: stack, heap, and method area. The method area contains a program (byte code). The program consists of a sequence of JVM instructions. Each instruction consists of a single-byte character and possibly operands. E.g. the goto instruction has an opcode 167 and performs an unconditional jump to the address specified by the operand.

The stack is used for allocating local variables, for storing parameters and results of JVM instructions, and for passing parameters and results when calling methods. Stack memory is allocated in stack frames. Each method call allocates a new frame. The frame contains memory for local variables and the so-called operand stack. The JVM maintains a pointer to the so-called stack top, which is the last value entered. The operand stack is used for JVM instruction parameters and results. Running program memory Ex. the iadd instruction fetches two values from the stack top, sums those values and places the result on the stack top.

Each operand stack item has a length of at least 4 bytes. Thus, all JVM arithmetic instructions work with operands of at least 32 bits in length. In the variable area, memory is also allocated after 4 bytes. That is, each variable occupies at least 32 bits in the JVM.A heap is an area of memory where objects can be created at runtime. The heap is created using the new keyword. Unlike C ++, for example, there is no need to delete objects. The so-called garbage collector takes care of the removal of objects in Java. The garbage collector itself detects unused objects and deactivates those objects. After the object is deactivated, the memory occupied by the object is ready for further use.

The description of the programming language is divided into two parts: syntax and semantics. Syntax describes the rules for correct writing. E.g. says where you need to write parentheses. If the program is syntactically correct, it can be compiled and run. Semantics defines the meaning of language constructs. E.g. says the = operator performs the assignment. For a program to do what we want it to be, it must be semantically correct. In this text we will discuss both syntax and semantics. We will discuss most of the syntactic constructs of Java and we will say its semantics. When writing the program we use so-called keywords. These are words that have a special meaning in the language (two of them, const and goto, but remain unused). Java is case sensitive. E.g. Class is something

other than class. Keywords are only lowercase.

1.3. BlueJ

BlueJ is a free, multi-platform development environment developed specifically for object-oriented programming in Java. It allows students to design a class diagram of a developed application in a simplified version of UML. The main advantage of BlueJ is its interactivity - it allows you to instantiate individual classes, send them messages and call their methods. (Wikipedia)

1.4. Installation

You must install Java first. Download the Java Development Kit (JDK) from oracle.com. BlueJ can be downloaded from BlueJ.org. Install both. Instructions for using BlueJ can be found, for example, at: https://www.bluej.org/tutorial/tutorial-english.pdf.

1.5. Creation of variables

Values are stored in variables. A variable is a named location in memory. In the framework of the creation of the variable - its declaration, in the Java programming language, it must first be stated with each variable what type of data to be included in Example 1 in our case int, boolean, char. Their meaning is explained in the next paragraph.

The name of the variable follows. The names of variables are written without diacritics and spaces. The spaces between the words are simply omitted and every other word starts with a capital letter. Close the term with a semicolon.

Example 1

```
int a;
boolean IsTrue;
char CarConsumption;
```

In Example 1, We created variables, but we did not assign any values. In Example 2, we created variables to which we assigned specific values. The first assignment to the variable is called initialization.

Example 2

```
int a=2;
boolean IsTrue =true;
char CarConsumption ='A';
```

If we want to declare more variables at a time, we must separate them with a comma:

```
int a, b;
```

We will distinguish between expression and command. The expression always has some value. This value is obtained by evaluating the expression. E.g. 42 and x + 1 are expressions. The command is the code that does something. E.g. X = 1; Is a command that assigns a value of 1 to the x variable. If we have an expression that evaluates something, we can usually command it by writing a semicolon behind it. In this case, we say that the evaluation of this expression has a side effect.

2. Data types

2.1. Integers

Int - We can only enter integers ranging from -2,147,483,648 to 2,147,483,647, including zero. An integer can represent, for example, the number of computer operations per second.

Other integer data types are listed in the table:

Type	Bytes	Range	
byte	1	-128	127
short	2	-32 768	32 767
int	4	-2147 483 648	2147 483 647
long	8	-9,22 x 1018	9,22 x 1018

If the result of the operation does not lie within the permissible interval of the given data type, an overflow occurs. In this case, the result is opposite to the result of the mathematical operation. It does not cause an error in Java, but it should be counted. E.g. If we have a value of 127 in a byte variable and add 1 to it, it will be in the -128 variable.

2.2. Real numbers

To work with real numbers, Java has two primitive data types: float and double. Both use the same representation method: the real number is stored as a triple sign, mantissa, and exponent. So we only store numbers approximately. Float type uses 32 bits: 1 bit occupies a sign, 8 bits of exponent, and 23 bits of mantissa. The double type uses 64 bits: 1 bit for the sign, 11 for the exponent, and 52 for the mantissa.

2.3. Conversion

Conversion of a value to another data type. E.g. Converting double to int. Some conversions are done automatically, eg int type to double type, others need to request, eg type long to int type. Do not need to write auto-conversion:

```
int a = 100;
 long a = b;  // Automatic conversion from int to long
```

If the conversion is required, we specify the target type in brackets before the converted value. E.g. Convert from double to int as follows:

```
double d1 = 5.85;
int i1 = (int) d1;
```

The result of this conversion will be the value that is in the entry of the original number before the decimal point (for nonnegative values it is the whole part of the number). I.e. In variable i2 the value will be 5.

```
double d2 = -4.99;
int i2 = (int) d2;
```

In variable i2, the value is -4. Recall that this is not the whole part of the number because the whole part of the number -4.99 is -5.

2.4. Logical

Boolean - Contains only true and false expressions. It is also called a logical variable. If we compare two numeric variables in a <b. Let us call them a and b. If and is actually smaller than b then the expression a <b is true and true is set to variable c. If true is not in c, false is stored

```
int a = 5;
int b = 6;
boolean c = a<b;
System.out.println(c);
```

In this example, the system writes "true".

The false is internally represented as 0, the value true as 1.

2.5. Character

Char - We can store one character in it. Before and after this character, there must be apostrophes (simple quotation marks). We can print characters from the Unicode table. The char value can be understood as the character index in the Unicode table.

```
char c ='H';
System.out.println(c);
```

2.6. Text strings

We use String to work with strings. Chain constants are written into quotation marks.

```
String s = " Hi how are you?";
```

Chains can be joined using the operator +.

```
String s1 = "jdk", s2 = "7.0";
String s3 = s1 + s2;  // Creates a string "jdk7.0"
```

Another value can be added to the string. In this case, the value is first converted to a string and then the two strings are merged.

```
int x = 42;
String s = "answer is " + x;
```

The example creates the string "answer is 42"

2.7. Comments

There are three kinds of comments in Java:

- one line - starts with // and continues to the end of the line

- Multiline - starts with / * and ends the characters * /
- Documentation - starts with characters / ** and ends with characters * /

Documentation comments are intended for processing by Javadoc, which generates documentation in HTML format.

```
/**
    * Main program class.
    */

  public class Main {
     public static void main( String[] args ) {
        /* Prints the answer */
        System.out.println( 42 ); // answer is 42
     }
  }
```

By the comments, we add additional information to the source text. The comment transporter skips, so they have no effect on the program run. We should comment, for example, on the importance of variables, unusual procedures and non-traditional algorithms. Thus, sites whose meaning does not have to be obvious to a reader at first glance.

2.8. Terminal input and output

In this chapter, we'll show you how to write to the screen and read from the keyboard. The output of the screen is the so-called Output Current (System.out). We use three of his methods: print (), println (), and printf (). The call below lists: Hi Baby

```
System.out.println( ” Hi Baby!” );
```

The print () and println () methods print out the value we specify in brackets after the method name (this is the value we call the parameter). It differs by adding the transition to a new line to println (), so the next call to print () or println () will print from the beginning of the line. You can use strings to associate with the + operator when printing.

```
int v = 200;
System.out.println( ”Car moved ” + v + ” km/h” );
```

A more elegant output offers the printf () method (so-called formatted output), which is similar to the C language.

```
int m = 6;
System.out.printf( ” African elephant weighs %d tons”, m );
```

The printf () parameters are a formatting string and a list of values. The format string can contain so-called output conversions that determine the shape in which the appropriate value is printed. Each conversion starts with a% sign. E.g. % D is the decimal integer output. When executing the command, the appropriate value from the value list is placed instead of the conversion. If the value is missing or does not match the output conversion, an error occurs.

A special case is conversion% n, which does not match any value in the value list. This conversion is reflected by moving to a new line. In MS Windows, a pair of characters \ r (carriage return) and \ n (line feed) is used to move to the new line. Unix systems use \ n. Conversion% n ensures that the correct transition to the new line is used, ie \ r \ n on MS Windows and \ n on Unix.

For real numbers, we have% f conversion. We can specify the number of digits after the decimal point (the default value is 6). E.g. % .2f prints two digits after a decimal point.

```
System.out.printf( ” Euler’s constant is about %.2f%n”, Math.E
);
```

Characters printed using the conversion %c.

```
char c = ’@’;
int i = c;
System.out.printf( ”Character ’%c’ Is in the Unicode table
in position %d%n”, c, i );
```

We use a conversion for strings %s.

```
String java = ”Java”;
System.out.printf( ” Our favorite programming language is %s%n”,
java );
```

For a read from the keyboard, we have the so-called "Input Stream" (System.in) in Java. Mostly we do not use it directly but through the Scanner class. This class offers methods for reading primitive types and strings. If we use the Scanner class, our program usually starts with the import command, which tells the translator where to look for the Scanner class. Before we first read, we create an instance of the Scanner class using the new keyword. To retrieve an int value, call the nextInt () method at this instance. import java.util.Scanner;

```
public class Read {
   public static void main( String[] args ) {
      Scanner sc = new Scanner( System.in );
      int x = sc.nextInt();
      System.out.printf( ” Readed value is: %d%n”, x );
   }
}
```

The nextDouble () method is used to retrieve the double value. Read the string by next ().

```
Scanner sc = new Scanner( System.in );
double d = sc.nextDouble();
System.out.printf( ”Readed number is: %f%n”, d );
String s = sc.next();
System.out.printf( ”Readed string is: %s%n”, s );
```

3. Operators

Numbers can perform common arithmetic operations in Java: addition, subtraction, multiplication, division, modulo (the remainder after integer division). Operators are used to writing operations. E.g. The addition is written using the + and modulo operator with the% operator. Thus, the x + 2 entry is a write of the addition operation. Each operator works with one or more values or variables called operands. Depending on the number of operands we can divide the operators into unar (one operand), binary (with two operands) and ternary (with three operands). The federal operator is eg ++ (increment) and the binary operator is eg = (assignment). The only ternary operator is?: (Conditional operator). The result of the operator execution is the value (we say the operator returns the value). E.g. Binary operator + (addition) returns the sum of its operands. The type of result depends on the operator and sometimes on the operands. For operators returning an integer value, the result is either int or long. E.g. If we add two int values, the result will be int, if we add two long values, the result will be long, and if we add two-byte values, the result will be int. Operator / (partitioned) returns integer shares for integer operands. E.g. 15/4 is 3 and -15 / 4 is -3.

If the value of the second operand is 0, an error occurs. If at least one operand is real (ie, float or double), the second operand is converted to real, and the operator divided will return the ratio of both operands. E.g. 4.5 / 3 is 1.5. If the second operand is 0, the real divide will result in some of these values: plus infinity if the first operand is positive, minus infinity if the second operand is negative or NaN (Not a Number) if the first operand is positive Infinity, negative infinity, NaN or 0. Operator% (modulo) returns the remainder after integer division. E.g. 17% 4 is 1. This operator is also defined for negative values, but it is different from mathematics. The sign of the result is always the same as the first operand sign: -17% 4 is -1, 17% -4 is 1, and -17% -4 is -1.

Operators can be chained, ie, multiple operators can be written. E.g. X + 2 * y is an expression that contains addition and multiplication operators. The ranking of operators evaluates operator priority (English precedence). E.g. In the expression x + 2 * y, multiplication and then addition is performed first because the multiplication operator has a higher priority than the addition operator. The other ranking order can be given by brackets: (x + 2) * y. If multiple operators with the same priority are used in the expression, the associativity of the operator (associativity) is determined by the evaluation order. E.g. In x - y - z we first make x - y and then subtract z from the result. We say the subtractor operator associates from left to right. Thus, the expression x - y - z has the same value as the expression (x - y) - z.

Some operators have the opposite, ie from right to left. An assignment operator is an example. The return value of this operator is the value of the left operand after the assignment. In the expression x = y = 1, first assignment is y = 1 and then the operator return value (in this case 1) is assigned to x. Therefore, the expression x = y = 1 is evaluated as x = (y = 1).

3.1. Operators of incrementation and decrementation

The Increment Operator (++) causes the value of the variable to be incremented by one. It can be written in two ways: prefixed and postfixed. In the prefix notation, the operator precedes his operand, followed by a postfix. In both cases, the variable is incremented, but the difference is in the return value of the operator. The prefix operator returns the value of the variable after magnification, the postfix operator value before magnification.

```
int x = 1;
int y = ++x;
```

In the expression y = ++ x, the increment operator first performs because it has a higher priority than the assignment operator. This will increase the x value by one. The return value of this operator is x after magnification, ie 2. This is used as an assignment operator. Y will be 2.

```
int x = 1;
int y = x++;
```

In the expression y = x ++, the incremental operator first performs. Its return value is the value of the variable x before magnification, ie 1. The value y is stored in the variable y.

The decrementation operator (-) causes the value of the variable to decrease by one. It is used similarly to the increment operator.

```
int x = 1;
System.out.println( --x );
```

3.2. Logical operators

Logical expressions can be joined together by logical operators. The logical operator has boolean operands and returns a boolean value. There are 2 logical operators.

The operator and && returns true when and only if both operands are true.

```
if( x == 0 && y == 0 ) {
     System.out.println( "x and y are equal to zero" );
```

```
}
```

The operator or || returns true when minimaly one from operands are true.

```
if( x == 0 || y == 0 ) {
    System.out.println( ” At least one of the numbers x, y
 is equal 0” );
 }
```

Both operators use the so-called abbreviated evaluation, ie the second operand is evaluated only if the value of the whole expression is not known after the first operand is evaluated. E.g. If the first operand has a false value in the logical product, the second operand is not evaluated and the value of the whole expression is false. Because these operators are often used in terms, they are conditional.

In addition to conditional operators, Java also has operators that always evaluate both operands. A & (logical product) is written and | (Logical sum). They can be used in the same place as conditional operators.

```
boolean b1 = x > 0 & y == 1;
boolean b2 = x <= 0 | y <= 0;
```

If we combine and and or, it is necessary to keep in mind that and has a higher priority than or:

```
if( x == 0 || y > 0 && z > 0 ) {
   System.out.println( ”x is zero or y and z are positive ” );
}
```

3.3. Assigning operators

We have already recognised one of the assigning operators, the operator =. Other assignment operators allow us to perform some arithmetic operation with the variable. E.g. The operator + = adds a second operand to the variable.

```
x += 5;
```

Other assignment operators are - =, * =, / =,% =. Each of these is a record of the corre-

sponding operation with a variable on the left. E.g. Operator% = performs the modulo operation:

```
    x %= 6;  // stejné jako x = x % 6
```

3.4. Priority of operations

All operators return a value that is the result of the operation. They have the same priority and associate from right to left. In the statement,

```
    int y = x + = 1;
```

the operator first performs the = = and only then =. The + operator = returns the value x after adding 1. This value is used as the value of the second operator of the operator =.

We can sort the probed operators by priority (from highest to lowest):

- Increment (++), decrement (-)
- casting
- multiplication (*), division (/), modulo (%)
- addition (+), subtraction (-)
- assignment operators (=, + =, - =, * =, / =,% =)

Priority tells us how strongly operators are weighing on operands. E.g. Casting has a higher priority than multiplication, therefore, in the expression (int) d * 2, the casting is done first and then multiplied.

```
    double d = 5.8;
    int i = (int) d * 2;  // same like ((int) d) * 2
    System.out.println( i );
```

The result will be 10.

Assign operators associate from right to left and arithmetic operators (addition, subtraction, multiplication, division, modulo) from left to right.

All binary operators evaluate the operands in the same order: first left and then right. This order is significant if the operand evaluation has a side effect.

```
    int x = 0;
```

```
int y = x + x++;
```

On the second line, the + operator is first evaluated. Its left operand is 0, the right operand is also 0, so the operator returns 0 and assigns it to y. When evaluating the right operand, the variable x 1 is increased (the evaluation has a side effect). If the order of the operands is changed, the value of 1 is assigned to y.

```
int x = 0;
int y = x++ + x;
```

3.5. Relational operators and equality operators

We use so-called relational operators and equality operators to write conditions. Relational operators are:

- less than (<)
- greater than (>)
- less than or equal to (<=)
- greater or equal (> =)

Equality operators are:

- equals (==)
- is not equal (! =)

4. Basic programming structures

4.1. If

The if and switch commands are used to branch out the program. The if statement lets you break a program by some condition. It starts with the keyword if followed by boolean in brackets. Boolean is an expression whose value is true or false. Then follows the command. When executed, the expression in brackets is evaluated and if true (the condition is met), the command is executed. In the example below, it is tested whether the variable x is zero. If it is equal to zero, the variable x is assigned a value of 2.

```
If (x==0) x=2;
```

We use relational operators and equality operators to write the condition.

The if statement can contain the else branch that is executed if the condition for if is not met. If the condition is not met and the other branch is missing, nothing will be done (will continue after the if statement).

```
if( x == 0 )
      System.out.println( ” Can not be divided by zero!” );
  else
      z=5/x;
```

If you want to write multiple commands, we will use a block. The block begins with the opening bracket {and ends with the closing bracket}. The block is a Java command, so we can use it everywhere there's a command.

```
if( x == y ) {
      x++;
      y--;
  }
```

The block limits the validity of the variable declaration. Each declaration is valid only until the end of the block in which it is listed. We say it is local in this block.

```
 if( x == y ) {
      int z = y;
```

```
    }  // This parenthesis terminates the declaration of the
   //variable z
  // Here you can not use z
```

We can use a boolean variable in parentheses.

```
  // In the month variable we have the serial number of the
   month
    boolean isMay = (month == 5);
    if( isMay ) {
       System.out.println( ”time for love” );
    }
```

To write the opposite condition, we use the logic negation operator (!).

```
   boolean isHere = true;
    if( ! isHere ) {
       System.out.println( ”Is not here” );
    }
```

If we need to branch off program eg. By the value of integer variables, we can use a concatenation of if statements.

```
  if( x == 1 ) {
       System.out.println( ”one” );
    } else if( x == 2 ) {
       System.out.println( ”two” );
    } else if( x == 3 ) {
       System.out.println( ”three” );
    }
```

4.2. Switch

Thus, if (as in the previous example) is chained together, we can sometimes replace it with the switch command. His entry begins with the key word switch. Behind it is an int or String type (or a type that can be converted to int) in brackets, and a block with any

number of case labels. For every case, there is a constant (or a constant expression, an expression whose value is known for translation), a colon and a sequence of commands.

```
switch ( x ) {
    case 1:
       System.out.println( ”one” );
       break;
    case 2:
       System.out.println( ”two” );
       break;
    case 3:
       System.out.println( ”three” );
 }
```

In execution, the expression is considered to be the keyword switch and its value is compared with the values given in case (in the order in which they are listed). Once commencing, orders commence. Execution ends with the break command. If the break command is missing, all commands are executed until the end of the switch command. The switch command can contain a default branch that is executed if no case label matches.

5. Cycles

Cycles are used to re-execute commands.

5.1. While

The while cycle begins with the keyword while followed by the condition and the cycle body in brackets. The body of a cycle is either a command or a block. Within the while cycle, the condition is evaluated and, if fulfilled, the body of the cycle is executed. Then the condition is reevaluated and, if satisfied, the cycle body has performed again, etc. If the condition is not met, the commands per cycle are continued. If the condition is not met at the start, the cycle body will not be executed once. The cycle while is, therefore, a cycle with a repeat count of 0 or more.

Example of syntax:

```
While(condition) {
// body
}
```

A concrete example

```
int x = 5;
while( x > 0 ) {  // Do until x is greater than zero
     System.out.println( x );
     x --;
  }
```

This example writes numbers from 5 to 1. It will, therefore, run 5 times.

5.2. Do while

The cycle into while begins with the keyword in which the body of the cycle is followed, followed by the keyword while and the condition, see the syntax example. This is done as follows: first, the cycle body is performed, the condition is evaluated and, if satisfied, the cycle body is re-performed, the condition is evaluated, etc. If the condition is not met, it continues after the cycle. The cycle body is always at least once do.

Example of syntax:

```
do {
// body
} while (condition);
```

A concrete example

```
do  {
System.out.println( x );
        x --;
} while (x > 0);
```

5.3. For

The cycle for has the form: for (*cycle variable, condition, command*) It is done as follows: initialization of the control variable cycle first, then the condition is evaluated and, if fulfilled, the body of the cycle is executed. Then the command is executed, the condition is reassessed, and if satisfied, the body of the cycle is repeated again, etc. Initialization of the control variable of the cycle is performed only once at the beginning. If the condition is not met at the start, the cycle body will not be executed once.

Example of syntax:

```
for (cycle variable; condition; command){
// body
}
```

A concrete example

```
int a;
  for( a = 1; a < 10; a++ ) {
     System.out.println( a );
  }
```

The cycle variable can be newly declared within the cycle. This declaration is valid only in the given cycle (ie in the header and body of the cycle). We say that the variable is local in this cycle.

```
for( int a = 1; a <= 10; a++ ) {
     System.out.println( a * a );
  }
```

Any of the parts that control the cycle variable, condition, or command may be missing. If the condition is missing, it is an endless cycle (the condition is still met). If no control cycle or command variable is given, no command is executed.

5.4. Break a continue

In the body of the cycle, we can use the break and continue commands. The break command immediately terminates the execution of the cycle.

```
int s = 100;
  while( s > 0 ) {
     int n = sc.nextInt();
     if( n == 0 ) {
        break;
     }
     s -= n;
     System.out.println( s );
  }
  // Here will continue after the break executed
```

The continue command terminates the execution of the cycle body and goes to the cycle condition.

```
int s = 0;
  do {
     int n = sc.nextInt();
     if( n == 0 ) {
        continue;
     }
     s += n;
     System.out.println( s );
     // here goes continue
  } while( s < 100 );
```

6. Static methods

So far, we have written the entire program into the main method. If the same sequence of commands had to be performed in multiple places, we had to repeat these commands. This can be avoided. The methods allow us to divide the code into logical units and reuse them. In this chapter, we will understand the method as a static method. We already know one static method. It's the main method. In addition to the main method, we can declare other methods in the class, such as the method for printing program information.

```
class MainClass {
   static void printInfo() {
      System.out.println( "Version: 1.0" );
      System.out.println( " Autor: 007" );
   }
   public static void main( String[] args ) {
      printInfo ();
   }
}
```

The declaration of the static method begins with the keyword static. (See example above) followed by the return type and method name. The return type can be any Java type. If the method returns no value, we will specify the return type as void. The method name usually begins with a lowercase letter. If the name consists of more words, we divide the words by writing the first letters of the other words. E.g. SpoctiPolomerCruzniceOpsane. It is not customary to use the underscore method in the name. Behind the name of the method is the parameter list in brackets. The parameter list is made up of variable declarations. The separator of declarations in the parameter list is a comma.

6.1. Calling methods

In the place where we want to perform the method, we will write the method call. The method call consists of the name of the method and the parameter list. The return value of the method will be the value of the expression that is given behind the return statement.

The order in which we declare the methods does not matter. You can also call a method that is declared later.

```
class MainClass {
    public static void main( String[] args ) {
        Scanner sc = new Scanner( System.in );
        int x = sc.nextInt();
        long factorial = countFactorial( x );
        System.out.printf( "%d! = %d%n", x, faktorial );
    }
    static long countFactorial ( int n ) {
        long fact = 1;
        for( ; n > 1; n-- ) {
            fact *= n;
        }
        return fact;
    }
}
```

In the void method, you can use the return statement without parameters. This is used for premature termination of the method.

```
// print rectangle a x b from @
static void printRectangle ( int a, int b ) {
   // The minimum rectangle side size is 2
   if( a < 2 || b < 2 ) {
      return;
   }
   for( ; a > 0; a-- ) {
      for( int i = 0; i < b; i++ ) {
         System.out.print( '@' );
      }
      System.out.pritln();
   }
}
```

The return statement may occur multiple times in the method. However, it will always be done once as the last command of the method. Its execution causes the method to terminate immediately.

```
static boolean isPrimeNumber ( int n ) {
   if( n == 2 ) {  // 2 prime number
      return true;
   }
   if( n % 2 == 0 ) {
      // Even number is not a prime number (except 2)
      return false;
   }
   int sqrt = (int) Math.sqrt( n );
   for( int i = 3; i <= sqrt; i += 2 ) {
      if( n % i == 0 ) {
         // We found a divisor, so n is not a prime number
         return false;
      }
   }
   return true;
}
```

7. Instance variables

We refer to instance variables as instance attributes or shorter attributes (instances of instances or arrays). Static methods are already known. They are declared using the static keyword.

Instantial methods are declared similar to static. Unlike static methods, however, their headers do not contain the static keyword. Instantial methods often work with instance attributes.

7.1. Array

The array allows you to work with multiple values of the same type. For example, to store twenty int values, we can either enter twenty variables or create twenty element arrays. In many cases, the array works more easily. Type the array type in Java using square brackets. The declaration of the variable type array of the int items looks like this:

```
int[] p;
```

Array type variable is a so-called reference. It will contain a reference (reference) to the array. The array declaration itself does not. You can create an array using the new keyword:

```
p = new int[6];
```

In this case, we created an array of six int elements. If we need a number of array elements, we will use the NameArray.length. In our case

```
p.length
```

The value of this variable is set when an array is created and can not be changed (read only).

```
System.out.printf( "array p has %d elemets%n", p.length );
```

We access the individual elements of the array using indices, which are written in square brackets:

```
p[1] = 5;
```

The indices begin always from zero, the first number will have index 0, the second number will have index 1, the third number will have index 2 etc. The validity of the index is checked at runtime. Using an invalid index will cause a runtime error. Once created, array elements are initialized to values whose internal representation is 0. For numeric

types, this is 0, for boolean it is false and for character char, it is a character at position 0 in the Unicode table. We use the for loop to read values in the field:

```
int[] p = new int[10];
  for( int i = 0; i < a.length; i++ ) {
     p[i] = i;
  }
```

We will do most of the list of field elements again with the for loop:

```
for( int i = 0; i < a.length; i++ ) {
     System.out.println( a[i] );
  }
```

The length of the field must be non-negative. An attempt to create a negative length field causes an error. Field creation can be combined with initialization. In this case, new is not used:

```
int[]numbers = { 3, 5, 6, 7};
```

The field size is given by the number of values in compound brackets. The array may be a parameter of the method and may also be a return type.

7.2. Multidimensional arrays

So far we have used one index to determine the element in the field. We call this one-dimensional field. Java allows you to declare and create multidimensional arrays. E.g. The two-dimensional field of int items looks like this:

```
int[][] p;
```

A multidimensional field in Java is a field of fields. The variable p is a reference to an array whose elements are one-dimensional arrays of integers. Create a field using the new keyword:

```
p = new int[2][3];
```

We access the field elements using indexes:

```
p[0][1] = 1;
```

The number of array elements is in the variable length of the field.

```
      System.out.printf( ”pole p má %d řádků%n”, p.length );
      System.out.printf( ”první řádek má %d sloupců%n”, p[0].length
   );
```

We use nested cycles when working with multidimensional arrays:

```
   for( int i = 0; i < p.length; i++ ) {
        for( int j = 0; j < p[i].length; j++ ) {
           p[i][j] = 1;
        }
     }
```

Multidimensional arrays can be created sequentially. The subfields can then have a different number of elements. Thus, the two-dimensional array is not necessarily rectangular.

8. Classes

A class is a form that describes a uniquely bounded data set (s) and operations above them. We use the class to create instances, individual objects that contain the data itself (over which the corresponding operations are called).

For example, let's have a car class, this class describes that the car has a tag, type, age and mileage, and contains the information operation, the object (), which lists the type, age and non-tag type. Using this teme agee - class - then we create individual instances, in real life, we would describe them as specific vehicles.

Each newly created instance (vehicle) in the program assigns data (brand, type, age and mileage). When we call the operator information later on (), this object (instance) will write us the message specific to the vehicle.

8.1. Class declaration

For the class declaration, we use the class keyword that is preceded by the access specifier followed by the class name. The class body itself is enclosed in a block (brackets). If the name is composed of multiple words, the first letter of each word is usually written in large letters (eg ListingInfo). We do not use underscore. In the class, we can declare variables and methods. Variables and methods can either be static or instantiated.

```
class Cat {
       int weight;  // instance atribut
       int age;
     void showInfo() {  // Instance method
        System.out.println( info );
     }
  }
```

We can instantiate from the class. The class is a temwagee that tells how objects will look, what attributes and methods they will have. In our case, each Cat instance will have two int variables. By declaring the Cat variable, we introduce a variable in which we can store a reference (reference) on a Cat class instance.

```
Cat  v;
```

Because class-type variables refer to objects, they are called reference variables. Each reference variable occupies the same space: 32 bits in 32-bit JVM and usually 64 bits in 64-bit JVM. On the other hand, objects of different types usually have different sizes. The size of the object is given by its attributes. We create objects using the new keyword:

```
V = new Cat();
```

After executing this command, the variable v will contain a reference to an instance of the Cube class. Attributes and methods are accessed using a dot. We call the method by using the method name and the parentheses:

```
v.info = 1;
v. showInfo ();
```

We can create any number of instances from one class. These instances are independent of each other.

```
Cat v2 = new Cat();
v2. showInfo = 2;
v2. showInfo ();
```

Instantial methods are used to perform operations over objects of the given type. E.g. In My class we can declare a method that will increase the value of the attribute x by 1:

```
class My {
   int x;
   void IncreaseA () {
      a++;
   }
}
```

Instantial methods, as well as class methods, can have parameters and return value. Parameters and return value are specified in the method declaration.

```
class My {
     int x;
     // Adds dx to x and returns a new x value
     int moveX( int dx ) {
        x += dx;
        return x;
     }
}
```

9. Access specifiers

We use access specifiers to specify access rights to individual classes, their operations, and variables. Their importance is above all in concealing implementation details that the user can not (not) see and possibly use.

Public ..... From any class.

Private..... Only within the given class, no access from the outside.

Protected..... From any class of the same package, or from the descendant of the class anywhere.

none (package-friendly) . From any class of the same package.

As an example, we have an employee who has his / her age and salary. It also contains the IntroduceYourself method

```
class Employee {
   public Employee (int age, int wage) {
       this.age = age;
       this.wage = wage;
    }
      private int age = 1;
   public int getAge () { return age; }
   public void setAge(int age) { this. age = age; }
   private int wage = 1;
      public int getWage() { return wage; }
   public void setWage(int wage) { this.wage = wage; }
   public void introduceYourself(){
       System.out.println("My age a wage are "  +
      age + "years "+ wage + "Euros");
    }
 public static void main(String[] args) {
      Employee employee = new employee (30,100);
   }
}
```

9.1. Heredity

Using heredity, we can create a class hierarchy in which we can tell any node that it is a special case of any of its ancestors. In the normal world, we would say that a chair is a type of furniture, an airplane is a type of means of transport. But we can not inherit the

chair from the animals because they also have 4 legs!

In Java, to create a subtype in the class header, immediately after the name, we include the extends keyword and the expanded class name. This class will inherit all non-private (including package friendly, if the extension class in the same package) methods and class variable ancestors that can be declared and overlapped again.

On the contrary, the class does not inherit the private and static methods of its predecessor, since they relate only to the particular class of the ancestor. End (final) methods marked as a descendant, the class inherits but can not overlap them. Each child object can be cast on an ancestor.

9.2. Super

When calling subtype methods, we often find that we do not want to overlap the whole method, we just want to add more functionality to it. At this point, we can call super.method (), calling the ancestor functionality. We can also call the ancestor constructor by calling super () - this must be the first call in the descendant constructor.

As an example, we created the class director, which is derived from the Employee Class. However, this does not mean that this class should have access to all private variables and methods of the original class automatically. See. Access specifiers.

Calling the parent class constructor is done using the super keyword and must be the first command in the constructor's body. If we do not call the constructor of the parent class, it automatically joins the program - in this case, the so-called implicit constructor, ie a constructor without parameters.

```
class Director extends Employee {
   public Director(int age, int wage)
   {
      super(age,wage);
         }
      public static void main(String[] args) {
      Employee k = new Director(30, 50);
        }
}
```

10. Polymorphism

We can overlay any method in a child's own implementation. When we call this method over a given object, the overlapping code will always be executed. Regardless of whether we approach the method by reference to an ancestor object or a descendant object (whose class contains that overlap).

This property is achieved by late binding, when the type of object on which the method is called will be decided at runtime, not during compilation.

This, however, is only a wager for overlapping methods, not for overloading them. If we have two methods on a given object that will differ only by the parameter (one for the ancestor, the other for the descendant), then the method that has the same parameter in the parameter as the current reference to the object is called. Calling overloaded methods is decided at the time of translation when it still does not have to be clear whether the reference will point to an ancestor or offspring object.

OBJECT-ORIENTED JAVA PROGRAMMING

1. Classes

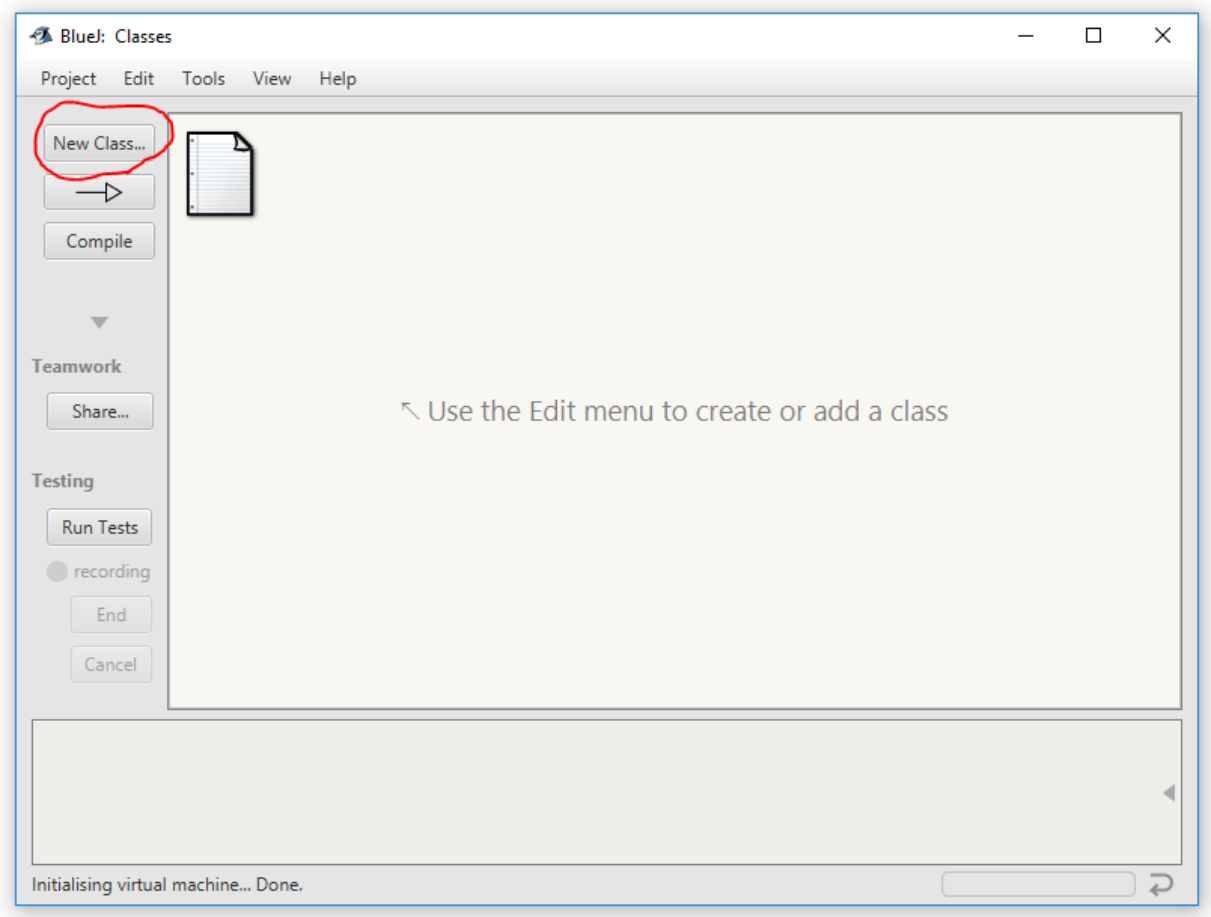
In real life, for example, the word chair is called a piece of furniture. Their function is to sit on it. It is a description of the object, which is characterized by various attributes or functions. An analogy in programming is a class. The class groups objects with some common properties. In real life, there are many different chairs that can vary in material or color. In programming a particular chair corresponds to an instance of the appropriate class, sometimes also called an object. Thus, the class can be compared to a form into which a particular thing is instantiated. Typically, classes can have any number of instances.

Individual objects can communicate with each other, send to each other different messages in which they can request different information or services. An example can be a calculator that we can ask for a lot of tasks, such as adding two numbers. Each of the functions of this calculator is professionally called the method. Requesting the use of a particular method is called a method call. Thus, the method is part of the program that an instance uses as a response to a method call (receive a message). The method builder defines how the object should respond to the message.

The entire object-oriented program Pecinovsky (2004) describes as written in a programming language a description of used classes, objects and messages that send these objects, supplemented with more complicated programs by describing the placement of programs on individual computers and assigning them to the administration of the respective service programs. (For example, OS or application servers).

1.1. Creating classes

Now we are going to build the classes themselves. Thus a kind of patterns, which produce by a specific instance (object). In the BlueJ program, which we will use throughout this course, we will choose a new project. Then select the button on the left. As shown in the picture below.

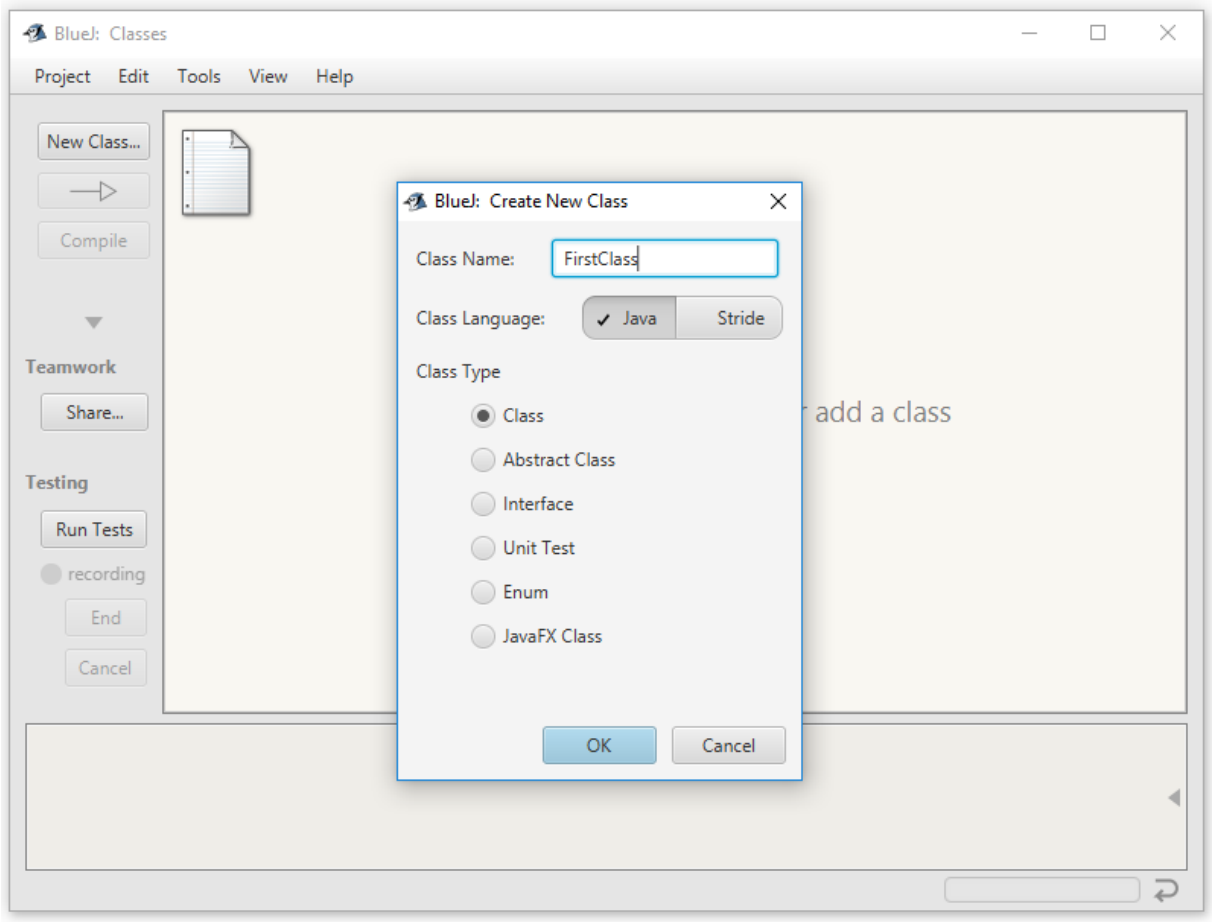


In the following window, we select the simplest possible definition of class - class. Next, choose the class name. In our project we have chosen the name FirstClass.

Class name is governed by several rules:

Rules for creating name identifiers
Links, classes, and others must be named for easy identification. We call these names as identifiers. These identifiers must meet several conditions:

- It can contain any characters that are included in a set of UNICODE
- Gaps must not be used
- It is customary to write multi-word class names without spaces. Individual words then begin with capital letters like: My First Pride
- Case and lower case letters are considered different
- The length is unlimited
- It must not start with a digit
- Can not match the keywords



After creation and translation, we have the first class. If we look into the folder where we have saved the whole project. We'll find that there are several files in the folder

FirstClass.java	We call this file as the source file. We will write a program describing the behavior of the class and therefore its instances. This file is a text file only. You can edit it in any text editor.
FirstClass.class	A translated byte code is stored in this file
FirstClass.ctxt	BlueJ additional file. If you delete this file, BlueJ will recreate it again at the next translation

1.2. Working with the class

After double-clicking on the class, we will open a text window where you can type or edit the class source code. See image below:

In the text window we will see:

- Class definition FirstClass
- Constructor FirstClass ()
- Method sampleMethod()

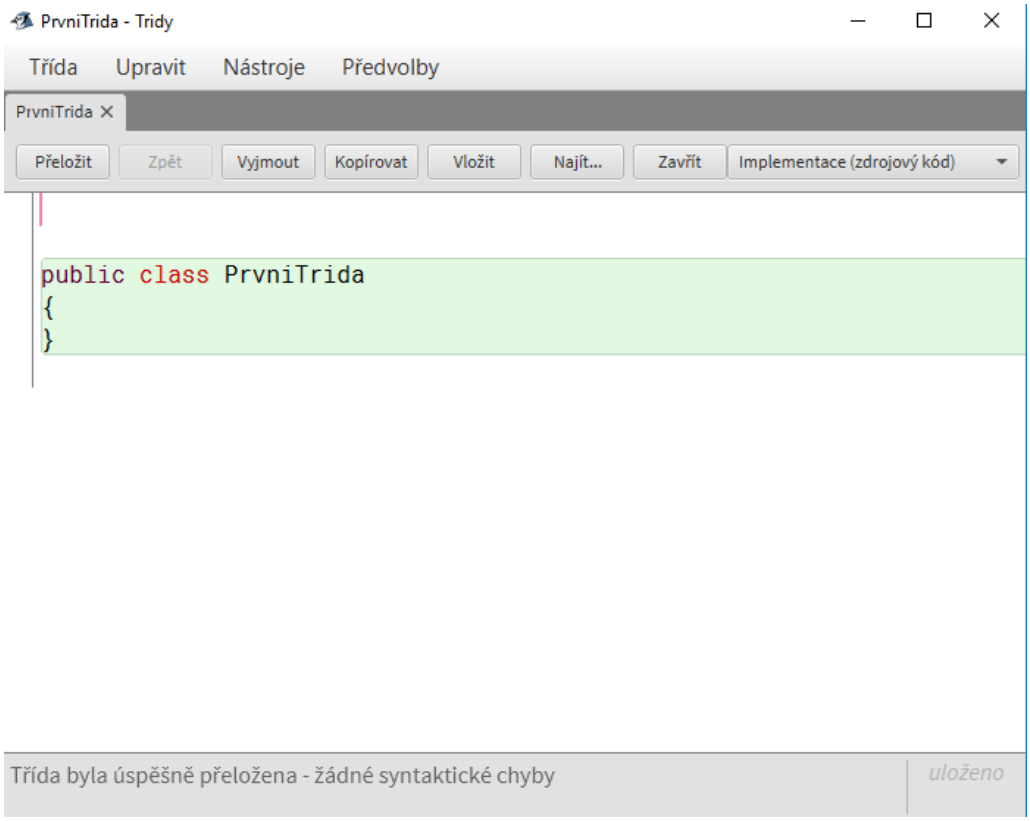
The class definition contains words

Public - A keyword that identifies that anyone can work with a class

Class - A keyword that indicates that it will be the class definition

FirstClass - Class name (its identifier)

The content of the class is contained in the brackets below. In our example, it does not contain any information, yet.



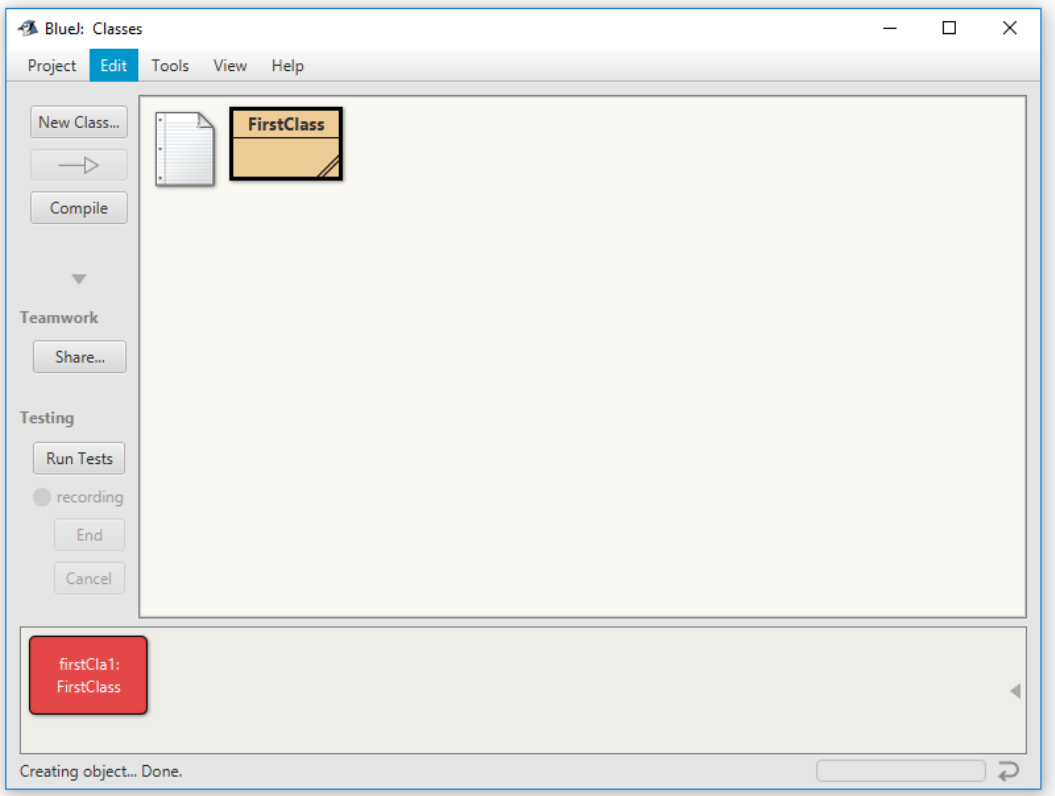
1.3. Creating a first instance

Close the text window. If you right-click on our first class. Attention! The class in question must be compiled! In the pop-up menu, select the new PrvniTrida () command. BlueJ will then ask us for the name of the instance being created. At the same time, we will offer it in front of the selected name. All you have to do is approve. Which is how we created the first class. As shown in the picture below.

By selecting the new PrvniTrida () command, we create an instance of the class by sending a message compiled from the new keyword, followed by the name of the class from which we want to create a pair of round parentheses. So we call a special method called a constructor. The constructor creates the requested instance and returns a link through which we will address the created instance.

In our case, we did not construct any constructor method. But the constructor must have every class! If we do not create a constructor, the compiler will automatically create the simplest constructor we designate - the default constructor. At the moment we create the first constructor in class, the compiler stops with the creation of the implicit constructor.

The created instance of the class is shown in the figure below. It's at the bottom, in the area we call the object bench. The instance of the class is red.



1.4. Remove a class

Very easy. Right-click on the class and choose Delete. Once confirmed, the class is actually deleted. However, the instance of the class itself remained. Requesting to delete a class is actually a request to delete the appropriate files from the disk. The class instance itself is stored in the memory. If we want to remove the instance of the class, we need to restart the virtual machine.

1.5. Restart the virtual machine

Restarting the virtual machine will delete all references in the link stack. So we delete all instances. We can do this either with the keyboard shortcut Ctrl + shift + r or by right-clicking on the rectangle at the bottom right and selecting the command: Restart the virtual machine.

2. Constructors

In the previous chapter, the compiler created an implicit constructor for us. Now we will show you how to create your own constructors.

2.1. Nonparametric constructor

Create a student class similar to the one we deleted. Now we create a nonparametric constructor. Nonparametric means that it does not require any information - parameters for its run. The class and constructor declaration itself could look something like this. The constructor itself is yellow:

```
    {
    int math =3;
    int english =3;
    int ICT =3;
    public Student ()
    {
            .........
    }
    }
```

Now we are going through the meaning of the individual parts of the text.

We first declared the Student and Variable math, english and ICT classes.

The constructor's header looks quite similar to the class header. The keyword keyword is omitted. The constructor name must be the same as the class name in which the constructor is listed. Here are the brackets into which the parameters are written, then the parametric constructor, if no brackets are written, it will be a parametric constructor. The body of the constructor itself is in brackets.

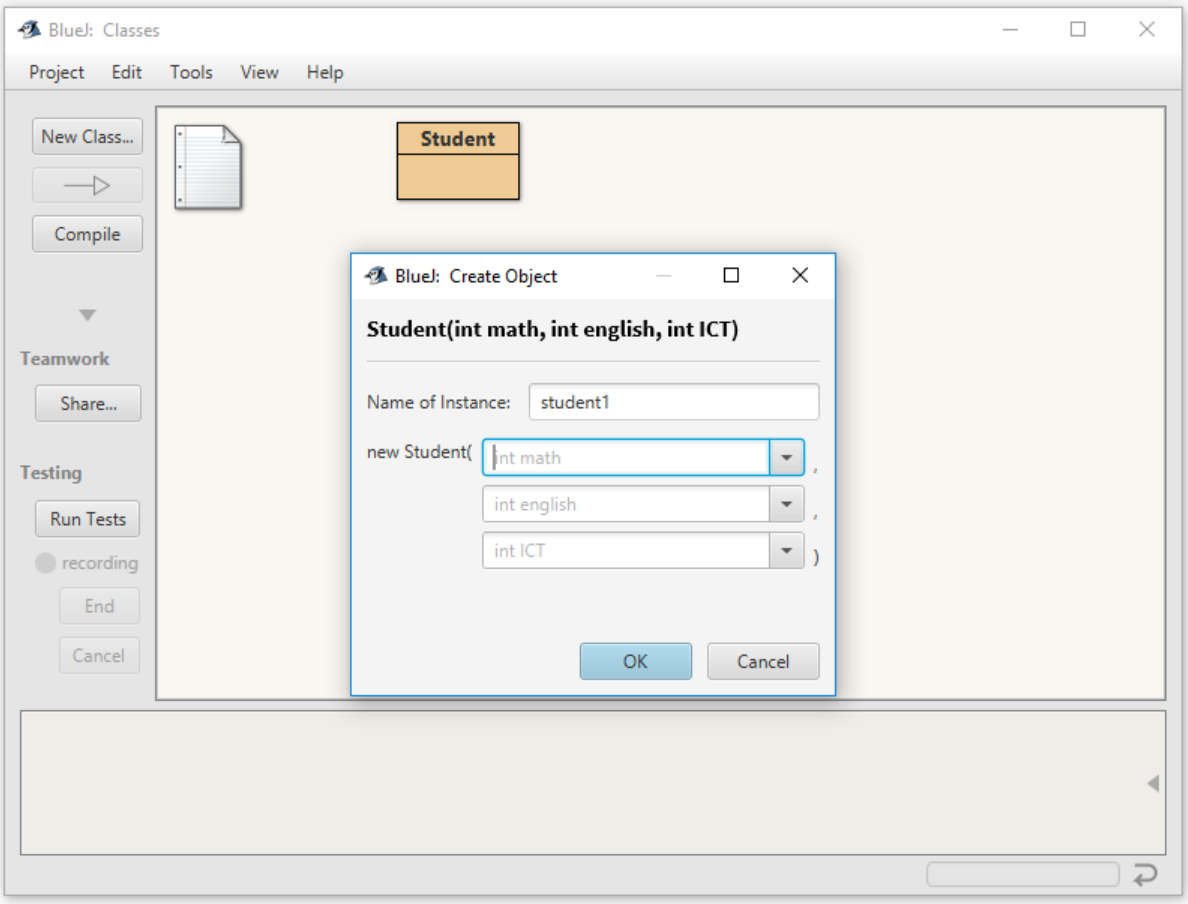
2.2. Constructor with parameters

If we were a Student instance, each student would have the same marks. Therefore, this time we create another constructor. This time with the parameters.

```
public class Student
{
      int math =3;
      int english =3;
      int ICT =3;
      public Student ()
      {
            .........
      }
      public Student (int math,int english,int ICT)
      {
            .........
      }
}
```

The new parametric constructor was highlighted in yellow. Both constructors have the same name. The translator distinguishes between them according to the number and type of parameters. To the parametric constructor, we first specify the type and consequently the identifier by which the program invokes the parameter in the constructor's body. Therefore, we can not construct constructors with the same types of parameters. The translator does not even distinguish the return values for individual constructors. The use of multiple constructors is called overloading.

Now that we want to create a new instance - a pupil using a parametric constructor, we will be asked to enter integer parameters as shown in the figure below.


BlueJ: Classes
Project Edit Tools View Help
New Class...
Compile
Teamwork
Share...
Testing
Run Tests
recording
End
Cancel
Student
BlueJ: Create Object
Student(int math, int english, int ICT)
Name of Instance:
student1
new Student(
int math
int english
int ICT
)
OK
Cancel

2.3. This

Many constructors are similar to each other. Often, we want to use another constructor in one constructor. This re-writing body constructor can be avoided by using this keyword, followed by a list of parameters. Beware of calling another constructor using this must be the very first constructor command! Using this constructor is another example.

```
public class Student
{
      int math =3;
      int english =3;
      int ICT =3;
      public Student ()
      {
            this (0, 0, 0);
      }
      public Student (int math,int english,int ICT)
      {
            .........
```

3. Methods

The method is a specific subroutine that performs a specific function. And one of the most commonly used tools (almost) of each programming language. We could compare the methods to the tools that each instance of the class is then equipped with. The method itself consists of several parts:

- **Access specifier** – which determines who can call the method. The most common are public and private. Specifier is optional
- **Type of return value** – Compulsory. If the method does not return, we writte void
- **Method name** – the same rules as in constructors
- **Method parameter list** – also the same principle as the constructors

The body of the method itself is enclosed in compound brackets. Where we can write individual commands. If we want, we do not have to write anything in the body.

```
public void Hello()
{
System.out.println(”Hello”);
}
```

3.1. Methods returning some value

If the method should return a value, we must specify its type in the header, and in the body of the method, we must return the statement followed by the variable we want to return. After the return statement, the method is immediately terminated. Therefore, the code that follows the return statement is not executed in the body of the method. The example is shown below.

```
public double averageGrade ()
{
double average= (math + english + ICT)/3;
return average;
}
```

3.2. Calling methods

By simply writing codes, no code will be executed. It will be done when we call it. You can

only call the method from the point where the method is accessible. Call the method by typing the method name and enter the parameters of the method into brackets. If the method is listed in another class. First we have to name the class, the method name is separated by a dot. If we send an instance message, we first need to write a reference to this instance. For attributes, the situation is similar.

3.3. Passing parameters to methods

Values of primitive types such as characters, logical values, or numbers are passed so that the value is copied to the local method variable

Object type values are passed through the link. Therefore, a link to the object in which the actual parameter is just copied into the local method variable.

4. Static attributes

Static elements belong to a class, not an instance. Static attributes are called static. Because it belongs to a class, all methods of that class have access to it. We can read static attributes even if there is no class instance. The static attribute declaration is listed below and is marked with yellow.

```
class Group {
private static int number = 15;

public void New(int count) {
number = number + count;
}
}
```

4.1. Static classes

Class methods are called in the class. These are often the help methods we often use, but we do not want to create an instance specifically for this purpose. As an example, a static method can be used to test whether a given number is plus.

```
public static boolean isPlus(int number) {
        if (number >= 0) {
                return true;
        }
        return false;
}
```

Attention! Because the static method belongs to the class, we can not access any instance attributes in it. These attributes do not exist within a class, but an instance.

4.2. Local variables

Sometimes we need to remember something in the method. Local variables are used for this purpose. We declare them inside the method. Beyond the method, they can not be accessed. This allows us to define another local variable with the same name in another method. We do not use access modifiers (such as public and private) or static in their declaration. We have to assign some value to their declaration. At the moment of leaving the method, the local variable is dropped. Therefore, there is nothing in them that we need between different methods. Frequent reasons for using local variables are to make the program more transparent and to reduce the number of errors that are caused by the repeated typing of complex expressions. The local variable declaration is as in the example below.

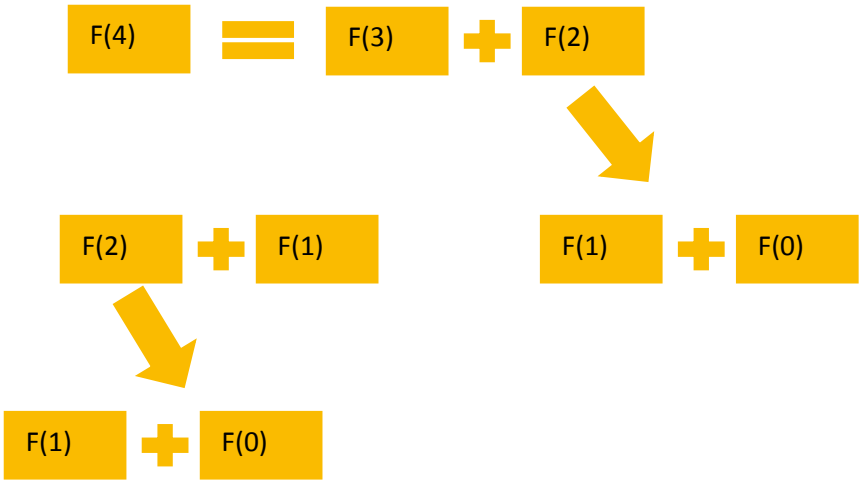
```
public void totalPrice (int pieces)
{
int totalPrice = pieces *15;
 }
```

4.3. Recursion

Recurring is defining an object (mathematically understood) by yourself. Recursive features must include some mechanism that ends the recursion at the appropriate time. If that did not happen, the recursion would go up to "infinity" The classic example of the end of the recursion is the insertion of stops.

A classic example of recursion is the Fibonacci sequence. For the Fibonacci sequence, each member of the sequence is the sum of its previous two elements. A F (0) = 0 and F (1) = 1.

An example of calculating the fourth member is shown in the figure below. It can be seen from Figure that in order to calculate F (4), we must first recursively recalculate F (3) and F (2). To compute F (3), we first have to calculate F (2) + F (1), and for F (2) we have to recursively recurs F (1) + F(0).



A big part of the calculation and recursion does repeatedly, making it computationally demanding. For this reason, it is often better to use classic cycles. In some cases, the whole algorithm is defined recursively (for example, Fibonacci sequence), or recursion may facilitate the work with some data structures.

A specific example of a recursive function call is shown below. In our example, we have 2 stops marked with yellow. The recursive call is then marked with a green color.

```
public static int fib (int n){
    if(n == 0) return 0;
    else if(n == 1) return 1;
    else return fib (n - 1) + fib (n - 2);
```

5. Encapsulation

Wikipedia (dated 7 May 2018) defines encapsulation:

Encapsulation is one of the fundamentals of OOP (object-oriented programming). It refers to the bundling of data with the methods that operate on that data.[5] Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them. Publicly accessible methods are generally provided in the class (so-called getters and setters) to access the values, and other client classes call these methods to retrieve and modify the values within the object.

Encapsulation is very important. If we had a complicated program. We could mistakenly influence the course, some of its parts. Finding and correcting such problems would take us a lot of time. Encapsulation is one of the basic concepts of object programming.

Encapsulation in Java is a mechanism for packing data (variables) and code. In encapsulation, the class variables will be hidden from other classes and can only be accessed by methods of their current class. Therefore, it is also known as hiding data.

We will achieve this by selecting the parts to be accessed by other features as public. This public part of the class. It is called the Class Interface. In the class interface, it is only appropriate to include what the other sections of the program really need to know about the class. For all the others we do not want to use the other parts of the program, set them private.

If some parts of the program will use some sections of the public class that will be subsequently changed. This may affect their functionality. Therefore, it is better not to change its publicly accessible parts after publishing the class.

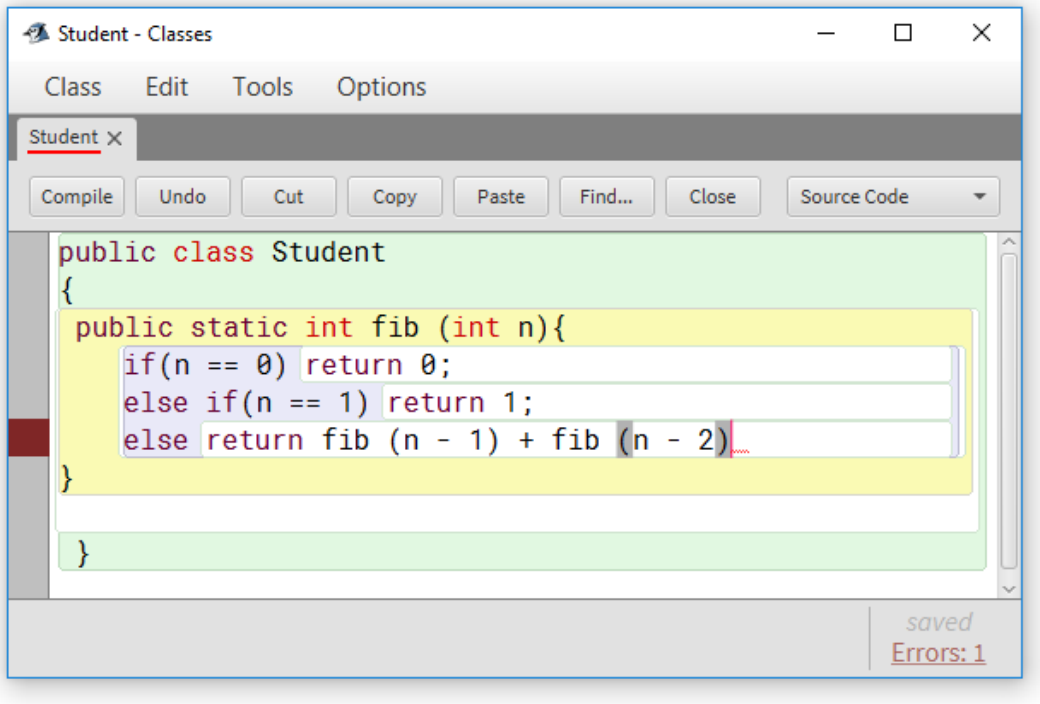
Often all class attributes are declared as private. At our discretion, we can publish access methods (setter, getter) that can be used to ensure that the attribute in question can only be read and not edited, or it can be set with some restrictive conditions. (For example, age is only a positive integer). If the attribute name matches the name of the parameter and I need to work with both, I use the keyword this. This word is used as the class name in which the method is included.

6. Debug the program

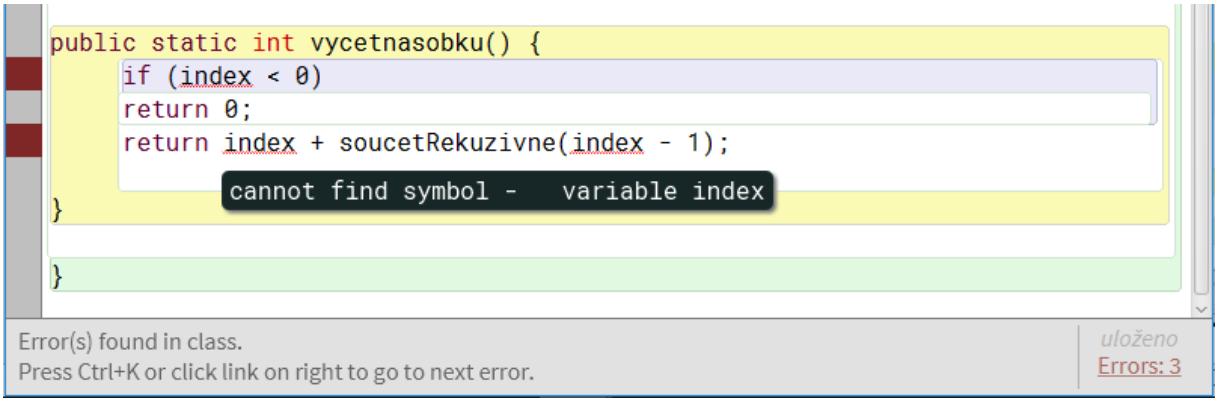
We often make a mistake when writing a program. These errors can be divided into several categories.

Frequent problems are syntactical errors. When do we overcome the language syntax? So against the rules of how to compose individual parts. Typically, for example, a forgotten semicolon or bracket is used.

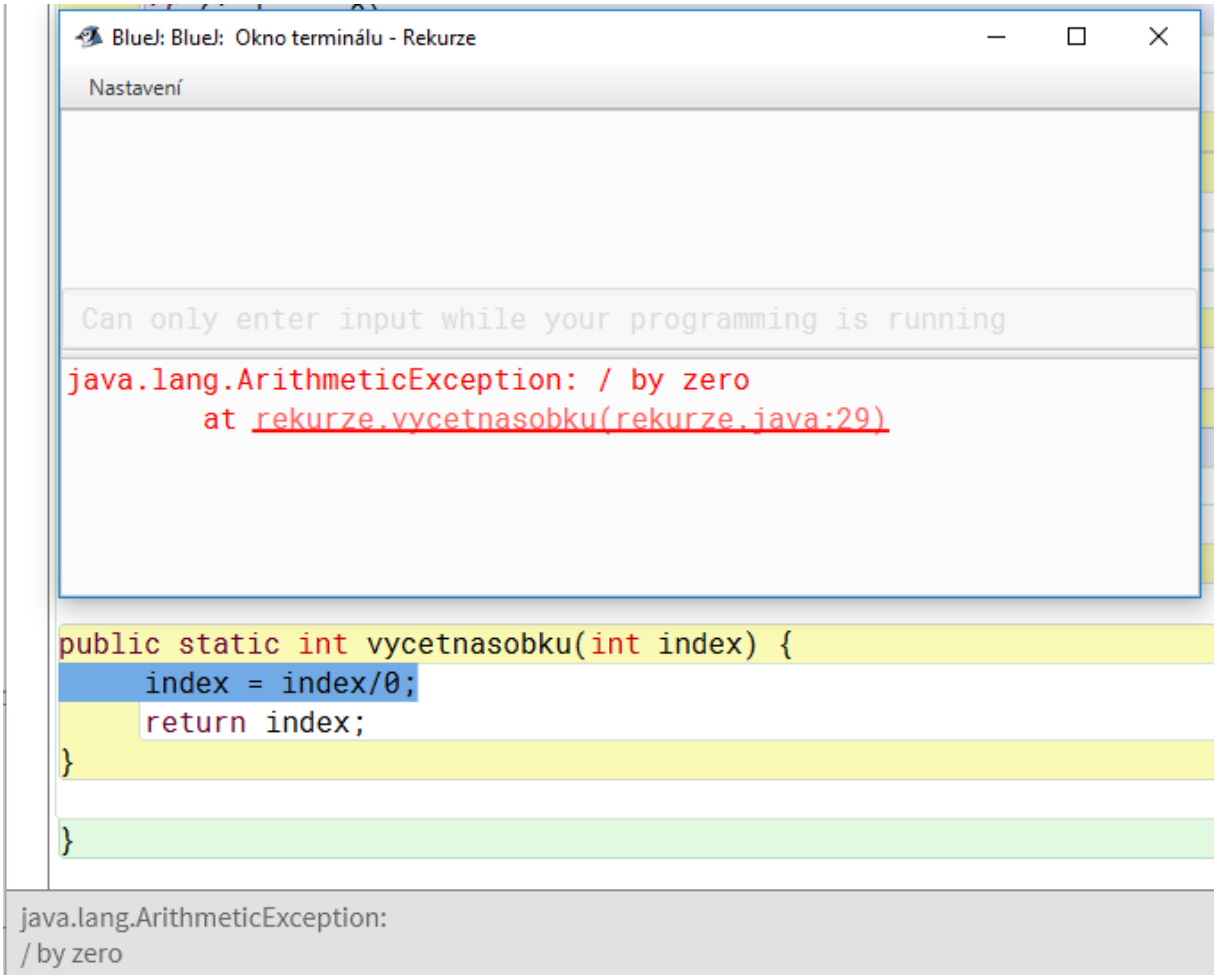
This problem is very often marked by the compiler before the compilation itself. At the same time, we will colorfully mark the line in which it assumes the problem. As shown in the picture:



Other errors appear when compiling. For more information on the error, we can jump to the bottom left of the word Errors. After clicking, we'll see more help for the error.



Other mistakes could be named as runtime errors. These errors are not found by the compiler. But at some stages of the program run they can happen. A typical example is zero division. Where the program stops when the zero break occurs, the terminal window opens in which it is displayed as the error is. BlueJ also marks the line where the problem occurred. As shown in the picture below. To remove such kind of errors. It is necessary to ideally try out all possible potentially problematic states of the program. So the user did not encounter similar problems. Ideally, we will prepare a set of test tasks in advance.



Semantic errors are the most insidious type of error. When the compiler does not appear, the program seems to work seamlessly. However, in some unexpected moments, the program shows an error. Which can be a big problem.

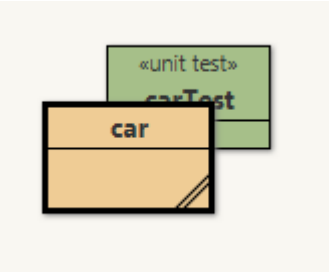
An example could be: Mars Landing Marine Crash (1999) Problem of communication between components - the user of the interface expected the value in kilometers, the provider gave it in miles. Instead of the planned 140-150 kilometers, it headed only 57 kilometers above the surface. At that height, however, Mars' atmosphere is too dense on the probe. Climate Orbiter burned at about 80 kilometers. (Source: Technet.cz)

This type of error is solved by software engineering.

6.1. Test Driven Development

In addition to testing the finished program, we can choose a different philosophy of software development - Test Driven Development. As part of this approach, we first define a set of tests and then we write the program itself, in which we only focus on passing the code through these tests. (We do not address code efficiency, for example) Refactoring follows. Duplicates are removed from the code, and the code is generally edited in the most acceptable form. Re-running tests will ensure that code functionality is not compromised during refactoring.

BlueJ offers a set of tools for this purpose. In BlueJ, we create a unit test for that class by clicking the class to be tested with the right mouse button and selecting the option to create test class. The given class then gets an inseparable partner - a unit test. The test class will differentiate colorfully. As shown in the picture.



If we want to perform the tests with the same set of objects. We can create a Unit test. We create the tool by creating only those objects that we want them to contain in the test tool in the object stack. Now right-click the test class and select **Object Bench to Test Fixture**. Note that all messages we have sent since the last virtual machine restart have been recorded. Therefore, if we want to be sure what is being done, restart the virtual machine first and then create objects. If we want to overwrite an already stored object with other objects, we simply create these objects and then choose **Object Bench to Test Fixture.** The product is overwritten.

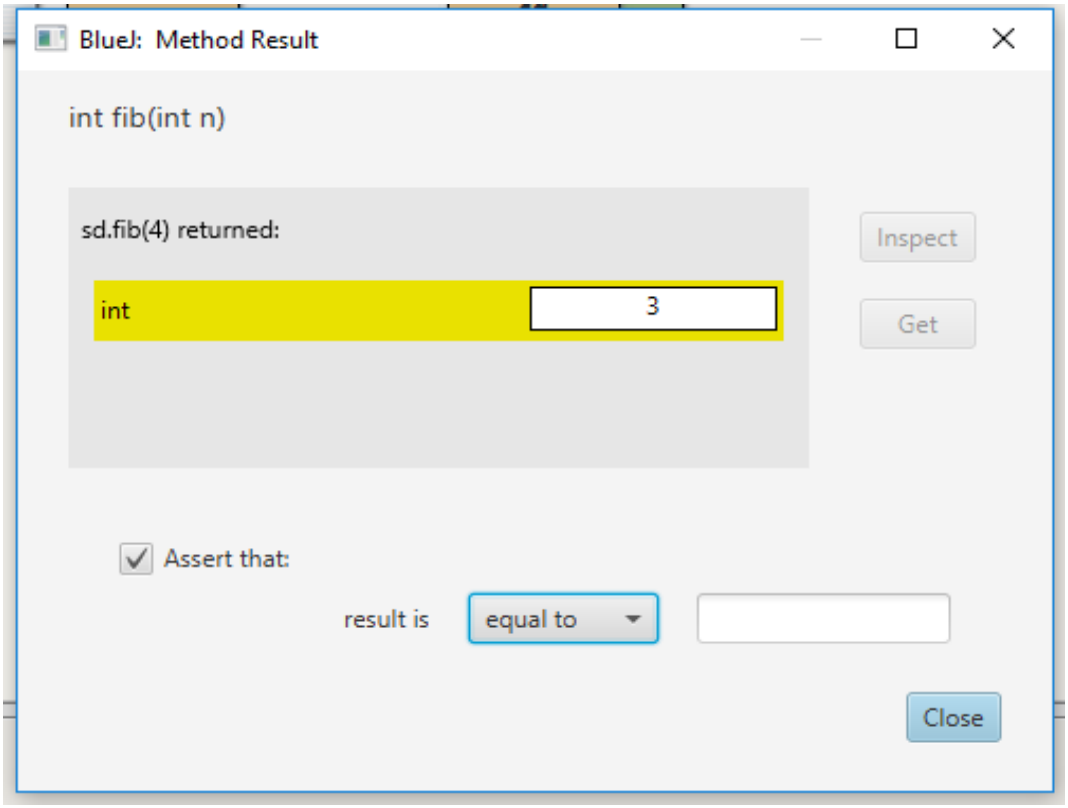
Alternatively, we can manually edit the test class.

To subsequently invoke objects, right click on the test class and choose **Test Fixture to Object Bench**.

6.2. Create a test

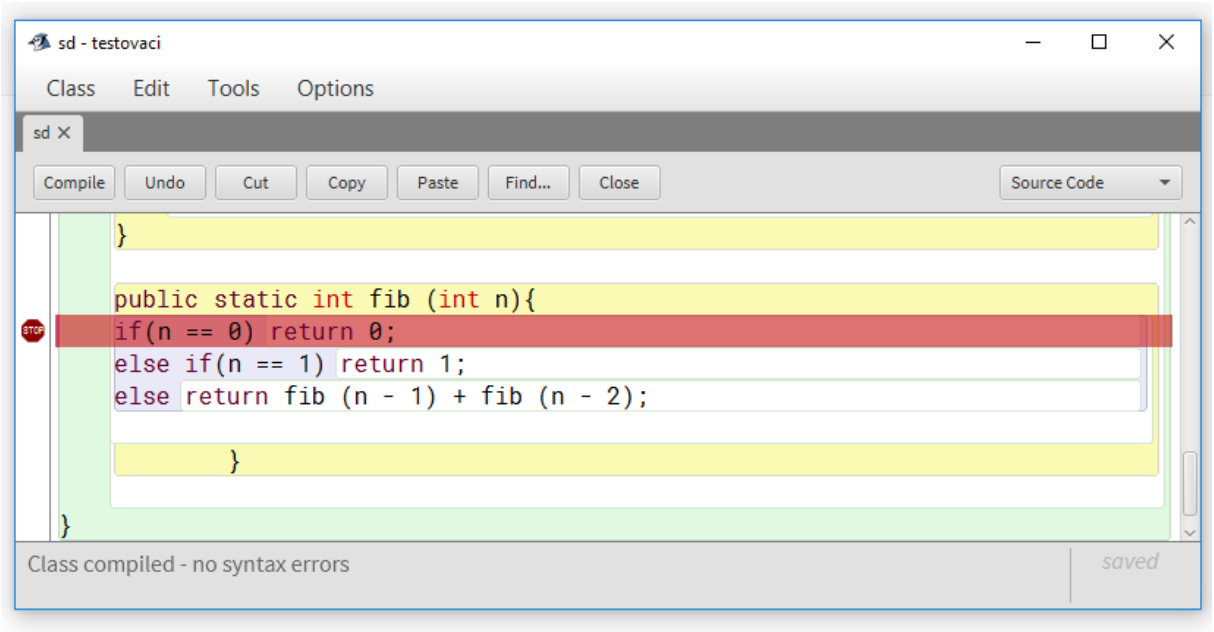
Ideally, we first restart the virtual machine. Now right-click the unit test and select Create test method. First, choose the name of the test method itself. Now BlueJ records our procedure for testing the product. We will do the required actions. During recording, BlueJ will ask us to test the return value for selected methods. It is shown in the picture below. During recording, the red light is on the left. If we want to stop recording without saving, we choose: Cancel. To end and save, we choose to exit. Both are just below the red button.

If we want to test the test, right click on the unit test, where we select the test in the menu. If we respond in the course of the test differently than the test return value. The test is interrupted. A test results window opens, where we can learn in which part of the code the error occurred.


BlueJ: Method Result
int fib(int n)
sd.fib(4) returned:
int
3
Inspect
Get
Assert that:
result is
equal to
Close

7. Debugger

It is a tool to help the programmer detect bugs in the program. To recall the debugger, we must first place a stop in the code. Which we will do by clicking on the left column in the code editor. On the appropriate line. The red stop mark appears here and the line is highlighted in red. As shown in the picture below. When the program runs, the program stops at the stop point and a debugger window appears.

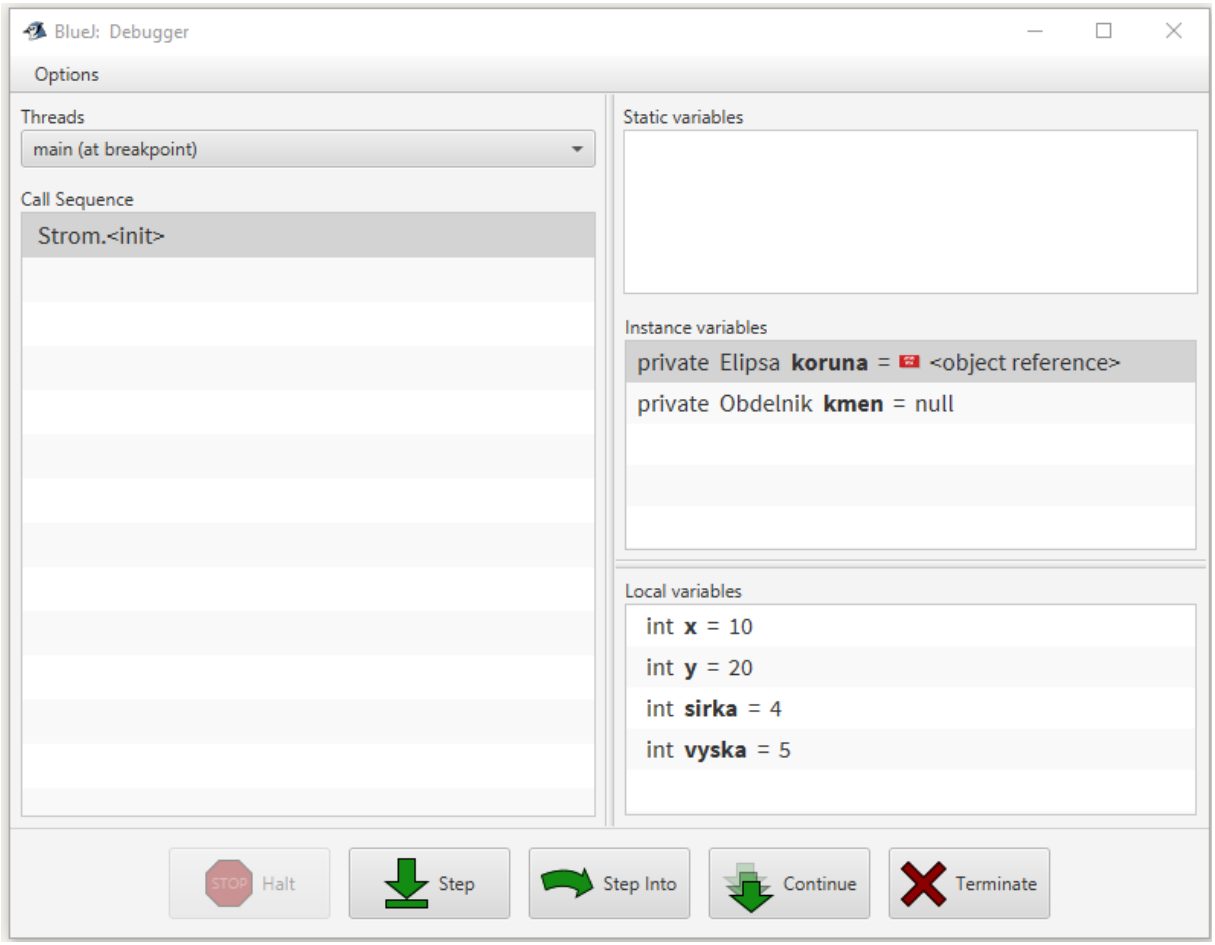


At the bottom of the debugger window (below), we have a total of 5 buttons.

- **Stop** to stop the program. It is useful, for example, if the program got stuck.
- **Step** to perform the next step of the code. The respective step is also graphically indicated by the green color in the code editor.
- **Step into** in is quite similar to stepping. The difference is that when you call the step method performs the whole method while stepping in, it will step step by step on the called method.
- **Continue** the program normally will continue until its end, or the next stop.
- **Terminate** to exit the program execution. Alternatively, we can end the program by restarting the virtual machine.

We can also add this new stops to the code during the process.

The debugger window itself is divided into several parts. In the top left, we can select a specific thread. Under the choice of thread, the area is Calling order. This area could be called a return address stack. All calls made during the program run are listed here. New calls are ranked highest. When we click on a specific call, the code editor opens with the class that contains the called method. Color is highlighted by the currently executed line.



In the right part of the debugger window, the **Class Attributes** are listed. These are class attributes that belong to the currently selected folder in the Call Order.

Below is **Instances Attributes**. The instance attributes that belong to the selected item in the Call Order are shown here. If we take a closer look at the instance attributes in our example, we find that this instance contains two attributes. The crown attribute contains a reference to the object. If you tap it, you will see a window with all the attributes of the primitive types of the instance. Attribute trunk at any given time has no link anywhere. (indicates null)

If the **Local Variables** also belonging to the currently selected item in Call Order are displayed below. The debugger here shows only variables that already have dedicated memory and the assigned start value, that is, they are defined.

8. Exceptions

Until now, we assumed that the whole program "will fall" as soon as any unforeseen events occur. However, this problem can be avoided by means of exceptions. In such situations, an exception will occur that immediately interrupts the run of the program and the transition of the thread to the location where the situation is treated. If that is not the case, the thread is terminated and since we only use one thread, the entire application will fall. There are several kinds of exceptions that can be broken down according to the cause of the unexpected situation:

- Error - critical error caused by, for example, lack of resources for the virtual machine - OutOfMemoryError, stack overflow - StackOverflowErrora similar. These exceptions are generally not treated.
- RuntimeException - often a bug of the programmer, (eg, zero division - ArithmeticException, invalid index - ArrayIndexOutOf BoundsException). To invoke this condition, we do not need to declare the ability to invoke the method in the header.
- Exception of the user (instead of a phone number entering letters), or a non-existent file or a file of another type. We can often very easily react to such types of exceptions. For example, let the user select the file once more. For such exceptions, we must always include the throws keyword and the list of called exception classes.

8.1. Protected mode

The potentially problematic part of the code can be placed in the "protected mode using try and compound brackets. This mode is a bit slower. However, if an error occurs - an exception, the part contained in the catch is executed. The optional block finally will always be executed. Finally, it is often used for cleaning jobs exceptions when we return to an exception, such as closing files, freeing up memory, and so on.

```
public static void exception () {
try {
int a = 5 / 0; //create exception
}
catch (Exception e)
{
System.out.println(”An exception was taken ”);
}
finally
{
System.out.println(”The contents of the
block will always be processed.”);
      }
 }
```

8.2. Throw

Alternatively, we can add the exception above using the throws keyword. This applies only to the exceptions checked, in case it wants to pass the exception treatment to the calling method.

As shown in the example:

```
if (index == null) {
throw new NullPointerException();
}
```

8.3. Throws

If we create a method that can create an exception we do not know or do not want to treat. We explicitly tell the translator that we pass this exception to the top level treatment using the throws + class exception

```
public static void  read ( ) throws IOException {
 ...
}
```

9. Working with files

Any information, data, or objects stored in the memory will be deleted when the program is shut down. To preserve and reload them, we need to save them to a file.

For trouble-free data entry, it is best to store application data in the appdata folder. The AppData or Application Data, or Application Data, contains data created by programs. In this folder, it creates its own folder virtually every program that is installed on the computer and then stores various data into it. The easiest way to get to this folder is to insert files into the roaming subfolder in the **%appdata%** file explorer, the data in this folder should follow the users on different computers within the domain.

The java.io package contains a File class that will provide us with all the important file-handling tools. Many methods from the File class require a file name as their argument. As an argument, you can use the String text string, or the instance of the File class. Which is often the most optimal solution that lets us find some information or perform an operation in advance about the file.

You can create a file object in several ways:

- The name of the file - we create from an absolute or relative path that converts to an abstract path.
- The name of the file relative to the parent - the abstract path will be created relative to the parental path.
- Uniform Resource Identifier (URI) - Certain requirements must be met. E.g. the path must not be empty.

For example, the file itself can be created using the URI in this way.

```
import java.io.File;
import java.io.IOException;
…

    public void createFile() throws IOException{
        File soubor = new File(”C:\\Temp\\hello.txt”);
        soubor.createNewFile();
                }
```

The file system may implement restrictions on certain operations on the current file system object, such as read, write, and boot, which we call access permissions.

Instances of the File class are invariant; this means that once the instance is created, the abstract path that the File object represents will never change.

The file class offers various methods for working with files. For example, we can work with:

- File Path - The File object is an abstract path to the file. This way we can work differently. Return methods returning the path to a file typically have a return value of the text string type For example, by using:
    - GetPath () gets an abstract path to the file
    - getName () find out the name of the file
- File Information - There are several methods that return file information such as: length (); canRead () or, for example, lastModified ()
- Directory Information - We have several methods for directory-only, such as list () - returning a list of all files in the directory or Unix SheetRoots () Returning Fields of All Directory Tree Root
- Working with files - We have different methods to work with files:
    - RenameTo (File k) as a parameter specifies the next instance of the File object.
    - delete ()
    - mkdir () directory creation
    - setReadOnly ()

10. Graphical User Interface (GUI)

It is the graphical environment the regular user encounters and works with. We all know very well the individual components of this environment. These include, for example, buttons, roller blinds, sliders, and the like. Different graphics libraries such as AWT (Abstract Windowing Toolkit), JFC (Java Foundation Classes) are available in Java. For BlueJ you can install Simple GUI Designer Extension. Which can simplify GUI creation. This paper will discuss how to work with JFC, also known as Swing.

10.1. First application

When creating an application with a graphical user environment in which the application will run. We must first create a "window" (frame). We will use the JFrame class. There are multiple options for creating a window. We created the ConstructGui class that inherits from the JFrame class. In this class we created a nonparametric constructor. We put a button and label in the class.

Another common component that we did not use in this example is the write field (JTextField). It would be declared as follows:

```
    JTextField someName = new JTextField(”Text”, 6)
```

As parameter enter the text that will be displayed in the field. Next, there is a number that indicates how many characters the de field should fit.

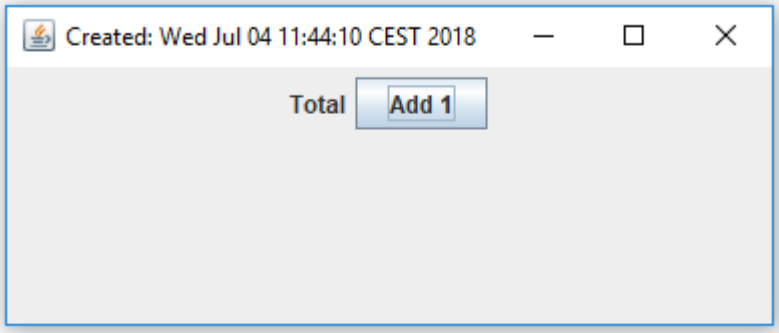
We have set the layout of components in FlowLayout. Due to FlowLayout, it was necessary to import java.awt.

The individual components needed to be added to the window. What we did using the **add ()** method, which specifies the name of the added object as the parameter.

```
import java.awt.*;
import javax.swing.*;

public class ConstructionGui extends JFrame {
    private JButton button;
    private JLabel value;

    public ConstructionGui()
    {
        FlowLayout layout = new FlowLayout();
        setLayout(layout);
        value = new JLabel(”Total:”);
        add(value);
        button = new JButton(”Add 1”);
        add(button);
    }
}
```

Afterwards we created another class with the name FirstGUI. With the method main. In the method main we created objects with the class ConstructionGUI. In this object we selected various basic methds. We imported a library of data, in order to use the function date(), which writes us todays date into the title window.

```
import java.awt.*;
import javax.swing.*;

public class ConstructionGui extends JFrame {
    private JButton button;
    private JLabel value;

    public ConstructionGui()
    {
        FlowLayout layout = new FlowLayout();
        setLayout(layout);
        value = new JLabel(”Total”);
        add(value);
        button = new JButton(”Add 1”);
        add(button);
    }}
```



The result looks like this:

The each components are centered. If we scale the window, they will be subordinated..

11. Events

At present, we have created a simple single-label window. However, nothing happens when the button is tapped. So we use events to add functionality to our window. The principle of events is that in the GUI we call for instance an event on the push button. The event we created the event is an event listener The method of doing something is invoked at the event listener. In our case, we create the Button and Button Counter buttons from the Label. The complete code is on the next page.

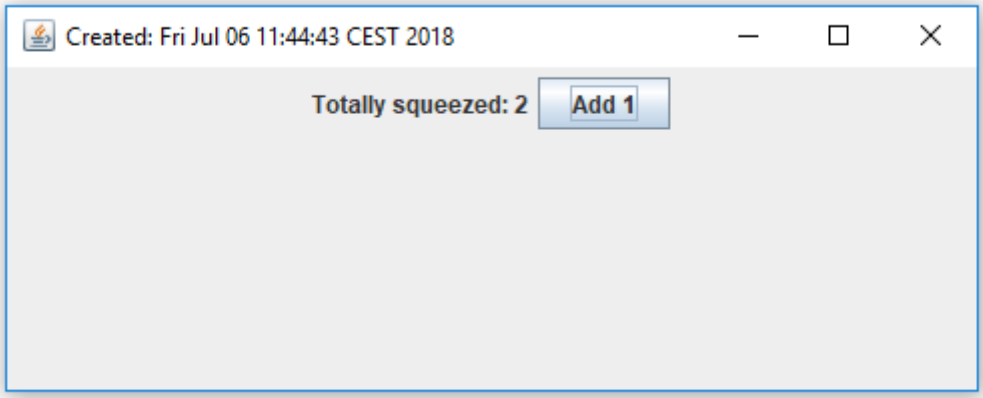
We added another class named EventCost to the ConstructionGui class. Thanks to this, it has access to ConstructionGui components. We have implemented ActionListener in the EventCost class. It is necessary to import java.awt.event. *;

The ActionListener class contains only one return action method actionPerformed (ActionEvent e) in which we define what happens when an event is invoked. In our case, counting is a push of a button.

In the constructor, we've created event listeners - an object named MyCount. We have assigned a button to the number of the listener.

```
EventCost MyCount = new EventCost();
MyButton.addActionListener(MyCount);
```

The result looks like this:



```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class ConstructionGui extends JFrame {
    private JButton MyButton;
    private JLabel MyValue;
    int MyNumber = 0;

    public ConstructionGui()
    {
        FlowLayout layout = new FlowLayout();
        setLayout(layout);
        MyValue = new JLabel("Not squeezed ");
        add(MyValue);
        MyButton = new JButton("Add 1");
        add(MyButton);
        EventCost MyCount = new EventCost();
        MyButton.addActionListener(MyCount);
    }

    public class EventCost implements ActionListener {

    public void actionPerformed(ActionEvent e) {

        MyNumber = MyNumber + 1;
        MyValue.setText("Totally squeezed: " + MyNumber);

    }
}
}
```

INTRODUCTION TO MEDIA STUDIES

1. Introduction into the media studies

Annotation: The central topic of the subject is the media communication as inseparable part of the modern and postmodern society. The introduction into the media studies deals with the basic stages of the development of human communication with accent on the role of mass communication and its effects on public – recipients of media information. Interdisciplinary issue is analyzed in relationships to its historical dimension.

2. Basic concepts – medium/media, mediation, medialization

Medialization: social change, the base of which is the unprecedented spreading of communication media and their all the time growing role in social life

Features of medialization:

- **extension**: media extend the possibility of human communication,
- **substitution**: media replace some social activities (television debate replaces the pre-election meeting),
- **amalgamation**: gradually, the borders between the media and non-media activities disappear – media definition of reality is mixed into one whole with the social definition of reality,
- **accommodation**: media are an important industrial branch.

Mediation: process, during which an intermediary enters between two parties to influence or to assure the relationship between them

Medium: mean; environment; it intermediates some action

Medium: (in the area of media studies) is an important link between the communicator and addressee, e.g. between the editors of newspaper, journal, radio or television, internet (e-mail or facebook) source and reader, listener, viewer, participant of internet discussion etc.

Media are means of mass or media communication transferring information in various forms and for various purpose, they may be subclassified in:

- **printed media** (newspapers, journals),
- **electronic media** (radio, television),
- **new media** (internet, social network -Facebook, Instagram, LinkedIn, etc.)

3. Communication, society, media, stages in the development of human communication

Basic stages in the development of human communication:

- Era of signs and signals
- Era of speaking and language
- Era of writing
- Era of print
- Era of mass communication
- Era of computers and network media

Marshall McLuhan (1911 – 1980) splits the human communication into the following periods:

- **period of oral tribal culture**: time of acoustic space
- **period of written culture**: acoustic perception is replaced by visual perception
- **Gutenberg galaxy**: a printed book is the first mass product, the first repeatable consumer goods type
- **Marconi galaxy**: period associated with the beginning of electricity

Today – in the period of digital computer networks, we can speak about **Gates galaxy**, but McLuhan does not write about it. **Werner Faulstich follows-up McLuhan and determines the periods of human communication as follows**:

- **Stage A: primary media** prevail: people function as medium till 1500.
- **Stage B**: **secondary (printed) media** prevail: 1500 – 1900 – printed media were at first individual and then mass matter
- **Stage C**: shift of focus on **tertiary (electronic) media**: 1900 – 2000 – mainly print, radio, later on television
- **Stage D**: shift of focus on **quartary (digital) media**: 2000 – up to now, trend from mass scope to individualization

In 1995, Mark Poster published the book „The Second Media Age", he splits the media development into two stages:

- **The first media age:** typical are: technologies of spreading (broadcasting) from one centre to peripheries with a high rate of integration and small rate of reciprocity (the word "broadcasting" is for him each spreading of type centre- periphery, i.e. also the distribution of newspapers and journals)

- **The second media age:** is based on the establishment of communication networks, the principle of spreading is replaced by the principle of interaction between the individual knot points of network with a low extent of integration and high extent of reciprocity

4. Typology of social communication

Initial parameters for distinguishing types of social communication:

- extent of individualization or socialization of communication acting,
- extent of institutionalization,
- number of participants,
- rate of relationship of equality/inequality.

Types of social communication

- intrapersonal communication (with himself/herself),
- interpersonal communication (between two or among three), the participants share the situation and communication context, role of speaker and listener disappear, the participants are perceived as individualities,
- group communication (in the group with a certain internal hierarchy – family, friends, a small working group, the authority enters the communication, manages it and the others respect it),
- intergroup communication (between/among formed groups: classes at schools, sport teams, higher extent of formalization),
- organizational communication (inside the organizational whole: form, school, political party), possibilities of dialogue are limited, relationship between the participants is not equal,
- Society-wide communication (all the communication processes, available to all the participants of a certain society), the dialogic character is lost, it is primarily uni-directional, individualization is weakened, the anonymity is strengthened.

The society-wide communication

Two special types are distinguished:

- Public communication (lecture, political meating, unity of place and time).
- Media communication (historically conditioned type of media communication is MASS COMMUNICATION).

5. Mass media, their characteristic, function and development

Characteristic features of mass media

- message is determined for a short-time use, they are of topical character,
- anonymous mass recipients,
- publicly available information for everybody without limitations,
- mainly unidirectional information flow,
- deferred feedback,
- periodicity of information,
- Information is offered regularly and continuously.

Function of mass media

- informing, providing information on events,
- socialization, explanation and commenting meaning of events, support of authorities and social standards, establishing the sequence of priorities,
- continuity, support of prevailing cultural formulas,
- entertainment, offer of excitement and entertainment ,
- acquiring, agitation for socially important goals.

Development of mass media

Beginning of mass media is connected with:

- with development of technical possibilities of production of a large amount of printouts of the identical content in the relatively short, known and regular period (publishing periodical print, projection of films, broadcasting programs).
- With social conditions for their use (the recipients change).
- With their economic appreciation.

Periods of mass media development

- beginning of the 19th century: development of mass press,
- beginning of the 20th century: film beginning,
- 20's and 30's of the 20th century: start of the mass radio broadcast (in our country 1923),
- 50's and 60's of the 20th century: beginning of the mass TV broadcast (in our country 1953),
- the last decade of the 20th century up to present time: establishment and development of digital media.

6. Recipients

Recipients: institutionalized collective user or recipient of some message.

Stages in the recipients development:

- Elite readership: formation of readership, low number of readers, educated
- Mass readership: it is formed in the 1st half of the 19th century from readership of gutter press and continues to exist up to the present TV-viewers
- Specialized readership: its formation was initiated by the development of art and science (specialized journals and scientific journals for interest groups, radio and TV stations)
- Interactive readership: with start of new media (internet and social networks)

The first media readership were readers. The invention of typography in the middle of the 15th century was the invention of record and possibility of multiple copying of the same message. The readers were exposed to a high number of copies of the same message. However, the readership was not numerous, the quantity of printing was limited, the books were expensive and their availability was low. This production did not allow the creation of mass readership – the mass spreading and periodicity of media offer was missing.

With the creation of the readership, also a new type of public comes into being: a part of nobility, burghers and the coming bourgeoisie started to be interested in their political assertion – the critically discussing public started to set up. The creation of public became a pre-condition for the establishment of the media and mass recipients, who needed media (printed materials), to look for the answers to questions they asked. Media address the public as one of its recipients group type and SIMULTANEOUSLY the public needs media to have a place where to discuss the topics in which it is interested.

Concepts of recipients

- Concept of passive recipients
- Concept of active recipients
- Concept of interactive recipients

7. Effects of media and their stages

- Thinking about the supposed effects of media, it is necessary to realize the basic influence: the influence of media on the individual is clear, but its explanation is difficult and the possibility of proof is not unambiguous and real.
- Moreover, it is necessary to realize that one and the same media contents may cause completely different effects, i.e. it may have a different impact on various individuals.

Criteria for the classification of supposed effects of media (as per Watson, 1998):

- What is influenced?
- In whom?
- To what extent?
- In what time span?

Based on these questions, we can distinguish the following effects of media:

- **Short-term and long-term ones**:
  - **immediate reactions** on media incentive – e.g the report on terrorism, fall of government, victory of a sportsman at the Olympiad,
  - **long-term changes** in the standpoints of individual or arrangement of society which may be caused a.o. also by media).
- **Direct and indirect ones**:
  - direct effects are very difficult to prove, but we can find them e.g. in the pre-election campaign of the candidate for president office and his subsequent success in the election; one of the direct effects is the influence of promotion campaign on the consumer behaviour of the customer, if the particular goods sale well after the promotion campaign,
  - indirect effects manifest themselves with a considerable time delay and parallelly with other factors.
- **Planned and unplanned ones**:
  - the planned effects are especially: commercial, political and social marketing, public relations,
  - the unplanned effects are e.g. the violence in the film broadcast by the TV and subsequent aggressive behaviour of the viewer etc.)

Character of effects of media:

- **Cognitive (recognizing): media offer incentives to learn –** e.g. geographic cognitive programs, programs about the world of science and technology, language courses etc..
- **Feelings :** especially the films and series evoking **emotion, kindness, tenderness, but also fear and stress**
- **Physiological:** when listening the music or following film, series, **relaxation may appear, as well as excitement and stress**
- **Behavioral:** these are first of all the changes in the consumer behaviour of the customer
- **Value (constructive or destructive): respect to weaker,** disadvantaged **persons**, effort to help them or on the contrary **effort for violent solution of conflicts**

In 1999, McQuail classified 1999 **four stages or research of media effects**:

- **Unlimited power of media (1900 – 1940**): direct effect of media contents, media contents arouse identical effect, i.e. they have identical influence on recipients
- **Non-effectiveness of media (1940 – 1965):** the individual personality characteristics distinguish and for this reason the individualized accepting media contents may occur
- **New faith in strong effects of media (1965 – 1980):** active attitude of the recipient to media
- **Transaction idea of media effect (since 1980 up to now):** strong position of media, but simultaneously a strong position of public

8. Media and power

Power (sociological definition): expression for a higher position in the social relationship (to force somebody to do something else than he wanted to do). If there is no possibility of constraint, we spoke about the influence very often.

John B. Thompson classified 4 **types** of execution of power in the society in the book "Media and fashionability. Social theory of media" (1995):

- **Economic power**: follows from the means creating the wealth
- **Political power**: follows from the elected or occupied position where it is possible to accept decisions
- **Coercive power**: follows from possibility of applying coercive power
- **Symbolic power**: follows from the possibility to create and to mobilize the support for power performance through the words, pictures or sounds

Relationship of power and media (2 attitudes):

- Media have so strong position in the society that their functioning may be considered for execution of power; media function within existing state formations, but are not economically connected with them. If the power elite is able to create and enforce the tools for control of media, it achieves such dominance over media and public that they will strengthen and consolidate the current power.
- In relationship of media and power, it is possible to find positive and almost healing features. Media act as watching dog of democracy, it means they execute the control power over the subjects and entities having the executive and legislative power (in our country: president, government and parliament).

Propaganda: persuasive form (persuasive communication), it is an intentional manipulation with thinking and behavior through symbols – it is a planned strategic communication. It has offensive character, it is long-term and conceptual, it strives for forming the world view, creation of desirable group or society-wide consciousness and models of behavior.

Types of propaganda:

- political: focused on keeping and acquiring of political power,
- economic: focused on purchasing and selling goods and keeping the trust in the economic system,
- war (military): demoralizing the enemy or support of moral of the own army and inhabitants,
- diplomatic: strenghthening of friendship (hostility) of allies (enemies),
- ideological: spreading of comprehensive systems of ideas,
- didactic: form of education of population, assertion of socially desirable goals,

- escapistic: specific form of political propaganda, which uses the media for distraction of attention from the social problems,

9. Typology of printed media

Printed media: media, the contents of which is bound to paper – leaflets, newspapers, journals, books etc.

Criteria of typology:

- Based on coverage of readership.
- Based on periodicity.
- Based on contents.

Based on coverage: printed media are distributed in a certain determined territory and are split into:

- **local media** (municipal newsletters, rural newspapers),
- **regional** (Českobudějovický deník/Daily from České Budějovice Region, Hlas Vysočiny/Voice of Vysočina),
- **multiregional** (Šumavský zpravodaj/Newsletter from Šumava),
- **nation-wide** (Lidové noviny, Hospodářské noviny/Economic Newspaper),
- **multinational** (Reader´s Digest).

Based on periodicity: periodicity is defined in the Act of Rights and Duties when publishing periodical press. Periodical press: are newspapers, journals and other printed materials issued under the same name, with the same focus and in the unified graphical layout at least 2x in calendar year.

- **Newspapers:** printed media (periodicals), being published minimally twice a week and containing a topical political part characterised by the manifold topics (diversity)
- **Journal:** printed medium issued in longer intervals than newspapers, maximally once a week and minimally twice a year.
- Collections of Acts and official bulletins shall be NOT CONSIDERED for the periodical printed media.

Based on contents:

- **weekly newsletters** (Týden/Week, Instinkt/Instinct),
- **social and life style (**Květy, Vlasta, Xantypa),
- **for children and young people** (ABC, Bravo, Dívka/Girl),
- **interests and hobbies (**Golf, Tennis, Cycling, Beekeeping).

10. Four theories of press

In 1956, four theories of press were formulated in the Article "Four Theories of the Press" by media analysts Friedrich Siebert, Theodor Peterson, Wilbur Schramm.

These are four types of attitude to solving the relationship between the society (political régime) and media.

- Authoritarian theory: media as a mean for announcing the standpoints and opinions of some authority (e.g. the ruler or politician owning media) which agree with the present power split
- Libertarian theory: is designated also as theory of free press. It prevents that the state controls the public communication - all medially presented opinions are balanced.Media are arranged in such a way that they are able to say at any time what they want.
- Theory of social responsibility: media should find various views of the given problem but there should be a limit, a boundary which shall be not passed (e.g. not to support the violence, crime, terrorism etc.)
- Soviet communist theory of media: media are tool of one type of socialization and forming public opinion, media serve as mean for education and public education (education of socialist citizen)

Denis McQuail suggested additional 2 conceptions:

- Developmental theory of media: media should contribute to the modernization of the society, they should achieve the socialization and educational targets.
- Theory of democratic participation: media represent the social institution, no democratic control of media shall exist, the media should be arranged in the way supporting the interests of minorities and individuals.

11. Broadcast as public service

The following television and radio organizations are among media with broadcast signal:

- of public sector (state, public – broadcast of public service)
- of private sector

Public sector in the area of broadcast media distinguishes by the degree of connection with the state machinery and government:

- **State radio and state television** are subordinated directly to the state machinery – in the countries with the authoritarian régime, they serve to this régime unilaterally and without reservations.
- **Public service radio and television** are established by the law; they are non-governmental organisation executing the non-profit public service.

Private sector in the area of sending media is organized on the base of private business; it appeared for the first time and exists up to now mainly in the USA.

In Europe, the radio broadcast started as licenced private business; as soon as the radio started to spread in a massive way, national states in Europe nationalized its broadcast. E.g. the Czechslovak stated entered by its majority in Radiojournal company in our country in 1925. Also private BBC (British Broadcasting Company) became a public BBC corporation in 1927 (British Broadcasting Corporation). !!! BBC became a pioneer: unlike other European radios, it did not become a part of the state machinery, however, it remained a public corporation independent on the state.

The public service models of broadcast developed in the western European democracies after the World War II, in the Eastern Europe only after 1989. The public service radio and television remained monopoly broadcasting subject in the given country – limits of oscilation spectrum frequency prevented the private broadcast, the role played also investment and operational demands of broadcast.

The development of private radio broadcast in Western Europe in the 70´s of the 20th century, the development of private television broadcast in the Western Europe in the 80's of the 20th century. – This means that in this time **dual broadcast system comes into being – the public and private radio and television broadcast exists simultaneously.**

12. Media education and media literacy

Media literacy: knowledge and skills, enabling to understand and to evaluate critically various aspects of media (media contents)

Media education: systematic hand-over of media knowledge.

Levels of media literacy:

- **Media education**: - **school** (see the cross-section topics of the General educational program for the elementary schools and secondary schools); - **extracurricular** (interest circles of young journalists).
- **Professional education**: - **education of teachers as well as future teachers** who are teaching or will teach media education; **education of journalists.**
- **Public education concerning media**: - **media criticism**: broadcast concerning media in the radio as TV; designation of suitability of TV programmes (e.g. the program is not suitable for children).

13. Important personalities of the Czech mass media

Josef Kajetán Tyl (1808 – 1856)

- writer, dramatist, journalist,
- in1833, he founded the journal *Jindy a nyní (At another time and now)* – he renamed it to *Květy české (Bohemian blossom)* – he wanted to create a Czech journal non-copying the foreign models,
- journal named *Květy* has been published up to now,
- effort for the public education and public activation of especially rural readership.

Karel Havlíček Borovský (1821 – 1856)

- 1846 - editor of *Pražské noviny (Prague Newspaper),*
- 1848 - he founded the first Czech daily newspaper *Národní noviny (National Newspaper),*
- 1849 - he founded the journal *Slovan /Slav/* (*Národní noviny* were stopped for political reasons), also here he criticizes publicly the Austrian political situation,
- 1851 - exile Brixen (Switzerland),
- 1855 back in Bohemia,
- representative of austroslavism: co-operation of Slavic nations within Habsburg monarchy,
- uncompromising critical thinking, logical arguments, cultivated Czech language.

Julius Grégr (1831 – 1896)

- Czech politician and journalist
- 1862 owner and editor of *Národní listy* newspaper; he succeeded to concentrate here the then elite of the Czech journalism – e.g. Jakub Arbes and Jan Neruda published in *Národní listy*

František Gel (1901 - 1972)

- 1924 – Lidové noviny (Folk newspaper) – editor,
- 1945 - editor of the political broadcast of Czechoslovak radio, reporter

from Nuremberg process,

- 1955 - Teacher of journalism at the Faculty of Arts of the Charles University.

Pavel Tigrid (1917 – 2003)

- writer, journalist, politician ,
- representative of the Czech anti-communist exile,
- 1939 emigration into England, he participated in the exile BBC broadcast in London,
- 1945 return to Prague, editor in chief of *Obzory (Horizons)* journal,
- 1948 again emigration, editor of Radio Free Europe broadcast, editor of *Svědectví* (*Evidence)* journal,
- after the year 1989, return into the Czechoslovakia again, president Václav Havel, ministr of culture.

IT SECURITY

1. Motivation

heise Security

« Vorige | Nächste »

Android-Schädling Tordow: Banking-Trojaner mutiert zum Super-Trojaner

28.09.2016 14:00 Uhr – Dennis Schirrmacher

vorlesen

(Bild: Christoph Scholz, CC BY-SA 2.0)

Sicherheitsforscher warnen vor einer Banking-Malware, deren Funktionsumfang sämtliche Träume von Kriminellen erfüllen dürfte: Der Trojaner kann im Grunde alles mit infizierten Android-Geräten anstellen.

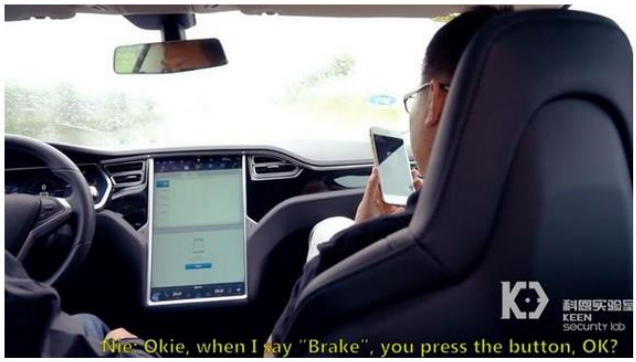
heise Security

« Vorige | Nächste »

Tesla Model S lässt sich von fern kapern

20.09.2016 08:24 Uhr – Andreas Wilkens

vorlesen



Screenshot aus dem Demo-Video der Forscher (Bild: Keen Security Lab)

Forscher des chinesischen Internetunternehmens Tencent demonstrieren, wie sie manche Funktionen eines Tesla Model S unautorisiert von fern steuern. Dabei gelang es ihnen auch, ein fahrendes Auto anzuhalten.

43 million passwords hacked in Last.fm breach

Posted Sep 1, 2016 by John Mannes (@JohnMannes)

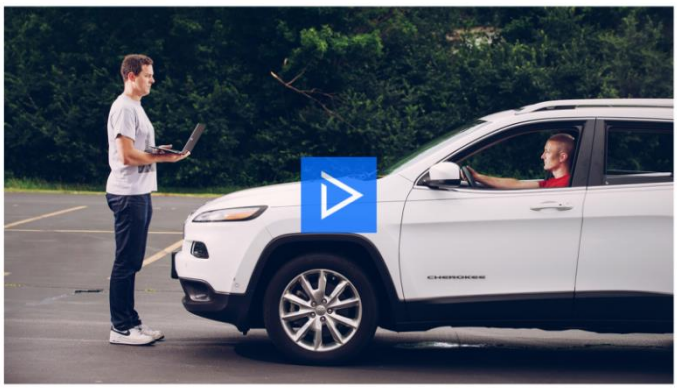
Crikey: 43,570,999 user accounts were breached in a hack of Last.fm that occurred in March of 2012, according to a report from LeakedSource. Three months after the breach, in June of 2012, Last.fm issued the following statement:

that Tordow's capabilities had significantly exceeded the functionality of most other banking ma

this allowed cybercriminals to carry out new types of attacks.

HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY—WITH ME IN IT

WIRED



Car hacking is the future - and sooner or later you'll be hit

theguardian

Security is finally being taken seriously but the fact that we are increasingly entrusting our lives to self-driving cars creates unease

A Tesla self-driving car on autopilot mode. Researchers explored the potential ways in which such vehicles could be hacked or exploited. Photograph: Bloomberg via Getty Images

DSGVO: Mehr als 1000 US-Portale für Europäer gesperrt

futurezone

08.08.2018

© Bild: Maksim Kabakou - Fotolia / Maksim Kabakou/Fotolia

Noch immer sind mehr als 1000 US-Nachrichtenseiten hierzulande nicht erreichbar, Instapaper ist jedoch wieder verfügbar.

foto: dado ruvic / reuters

Der Facebook-Hack sorgt weiter für Aufregung.

Weniger als zehn Prozent der 50 Millionen weltweit – Unternehmen verspricht "bald" neue Informationen

Von den fast 50 Millionen von einem Hacker-Angriff betroffenen Facebook-Nutzer stammen weniger als zehn Prozent, das sind maximal 5 Millionen, aus der Europäischen Union. Das teilte die zuständige irische Datenschutzbehörde am Montagabend bei Twitter mit. Facebook habe zugesichert, "bald" ausführlichere Informationen liefern zu können, hieß es in der knappen Stellungnahme weiter.

DSG: Verwaltungsstrafe bis EUR 25.000,-- DSGVO: Geldbußen bis EUR 20.000.000,--

https://www.dataprotect.at/info/geldbußen/österreich/

Auch bereits bei **Geltur**
Datenschutzverletzun
Verstöße) oder bis zu **E**
51: Datenverwendung i
Der **Strafrahmen für D**
4 % des weltweiten Ko
weltweiten Konzernum
absolute Obergrenze d

ENTSCHEIDUNGEN ZUM DATENSCHUTZRECHT · 20. September 2018

erste Geldstrafe durch die DSB in Österreich

Die DSB hat die erste **Geldstrafe** verhängt.

EUR 4.800,-- für eine nicht korrekt gekennzeichnete Videoüberwachung, die den öffentlichen Raum mitüberwachte.

https://www.dataprotect.at/2018/09/20/erste-geldstrafe-durch-die-dsb-in-österreich/

Nach Medienberichten (Salzburger Nachrichten, 19.09.2018) wurde gegen einen Betreiber eines Wettlokals in der Steiermark eine Geldstrafe von EUR 4.800,-- verhängt, weil eine **Videoüberwachung nicht ausreichend gekennzeichnet** war und ein **großer Teil des Gehsteigs von der Anlage mitaufgezeichnet** wurde. Die Überwachung des öffentlichen Raums in dieser Art und Weise, nämlich großflächig durch Private ist nicht zulässig.

- Strong dependence of modern society on ICT (especially critical infrastructures).

- ICT is becoming the Critical Information Infrastructure (CII)

- The following trends make the protection of ICT a central issue (according to Eckert2008, among others):
    - Globalization
    - Mobility (Smartphone, Information at your fingertips)
    - Networking
    - Ubiquitous/Pervasive Computing, Internet of Things (IoT)
    - Industry x.0
    - Smart Home/Grid/Car/*
    - Cyber War/Warfare/Espionage/*

- Conclusion: Protection of ICT is of central importance for society!

2.Protection objectives of information security

Information security (or IT security) has the task of ensuring the following protection goals (basic values) [Stallings2006]:

1. Confidentiality
2. Integrity
3. Availability

C-I-A

4. Authenticity
5. Accountability or non-repudiation

2.1. Confidentiality

- The protection of data from unauthorized disclosure. [Stallings2006]
- **Traffic Flow Confidentiality** = Protection from traffic data analysis
- Confidentiality in daily life
  - Confidentiality of Letters & Telecommunications Confidentiality
  - Official secrecy
  - Confession secret
  - Duty of confidentiality of attorneys/notaries/doctors
  - Non-disclosure agreements/NDAs
- Accessible by encryption (Encryption)
  - Symmetric encryption
    - a common shared key (same key for encrypting and decrypting) => key distribution problem; stream/block ciphers, substitution & transposition
  - Asymmetric encryption
    - public and private key (public key for encrypting and private key for decrypting) => key distribution problem solved, algorithmically

more complex and therefore slower

- Practice hybrid: Exchange of a generated session key via asymmetric crypto and then symmetric encryption

2.2. Integrity

- The assurance that data received are exactly as sent by an authorized entity (i.e. contain no modification, insertion, deletion, or replay). [Stallings2006]
- Changes (e.g. due to transmission errors or active attacks) **cannot** be **prevented** but can be **detected**!
- Authenticity and integrity are often seen as a unity. The one is useless without the other.
- Accessible through the use of (**cryptographic**) **checksums**, e.g.
  - CRC (Cyclic Redundancy Check) only for transmission errors
  - (Keyed-Hash) Message Authentication Code (MAC, HMAC)
  - Digital signatures

A **cryptological hash function** or **cryptographic hash function** is a special form of hash function which is collision resistant or a one-way function (or both).

A hash function is a function that maps a string of any length to a string of fixed length. Mathematically this function is not injective and not necessarily surjective.

2.3. Availability

- Availability is the property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system. [Stallings2006]
- What types of performance?
  - Usable or applicable
  - Sufficient capacity
  - Clear progress and/or clearly defined waiting times
  - Completion within acceptable timeframe
  - ...
- Approaches to the implementation/guarantee of availability

- Technical: fault tolerance and redundancy through e.g. RAID, clustering, ..., mirrored data centers, uninterruptible power supply (UPS), ...
- Organizational: Service Level Agreements (SLAs), Availability Classes

- Availability classes
  - 2-6: 99% (failure of ~ 90 hours per year)
  - 99.9%, ... to 99.9999% (= failure of ~ 30 seconds per year!)

2.4. Authenticity

- Authenticity
  - **Entity authenticity** = I know who I'm communicating with
  - **Data origin authenticity** = I know who the data comes from

- Authentication and authentication

- The assurance that the communicating entity is the one that it claims to be. [Stallings2006]

- Authentication features = feature with which a participant can be authenticated
  - Based on knowledge (PIN, password)
  - Based on ownership (key, card)
  - Based on property (biometric characteristics such as fingerprint, iris, voice, signature, retina, ...)
    - Retina = Back of the eye (infrared scan)
    - Iris Scan (using normal optical camera)

- Access Control (Access Control)
  - Authenticity is a prerequisite for access control!

2.5. Access Control

- The prevention of unauthorized use of resources. [Stallings2006]
- The **authenticity** of the accessing entity **must be ensured**.
- The following is checked
  - Who can have access to a resource,
  - the conditions under which the access may take place, and
  - what rights the accessing entity may have.
- Principle of necessary knowledge (**Need-to-know principle**)
- Basic models
  - Discretionary Access Control (DAC)
    - Definition of rights exclusively on the basis of user identity
  - Mandatory Access Control (MAC)
    - Not only user identity but additional rules and features
  - Role Based Access Control (RBAC)
    - Role-based assignment of rights, i.e. not on the basis of user identity but on the basis of user role (group membership)

2.6. Accountability or non-repudiation

- Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication. [Stallings2006]
- **Non-repudiation of emission** = Sender cannot deny sending the message.
- **non-repudiation of receipt** = recipient cannot deny receipt of the message
- Accessible through the use of digital signatures (not through [H]MACs!).

3.Information vs. IT Security

- Information Security (InfoSec) deals with **data in any form** (electronic, written, verbal).
- IT Security (IT-Sec) focuses exclusively on **electronic data processing**.
- Definition of IT security (transferable to InfoSec, according to [BSI1992]):

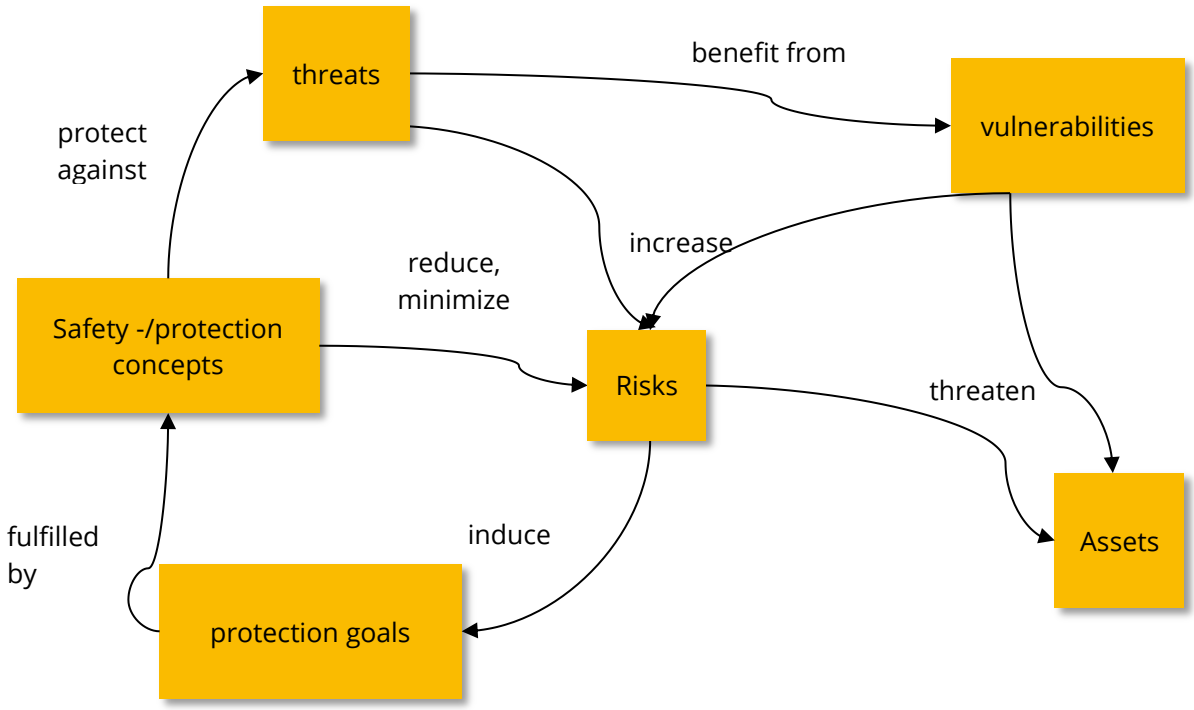
> IT security is the state of an IT system in which the **risks** that are present when using this IT system due to **threats** are limited to a **tolerable (acceptable) level** by **appropriate measures**.

- Highly important (!!) consequence: There is no absolute safety!

4. Weakness, threat and risk

- A **weakness/vulnerability** is a weakness of the system or a point at which the system may be vulnerable (through an **exploit**).
- **Threats** result from possible attacks that exploit one or more vulnerabilities in a system to compromise one or more protection goals.
- The **risk R** of a threat is the **probability E** of the occurrence of a loss event and the **amount of the potential loss S** that can result from it: $R = E \cdot S$

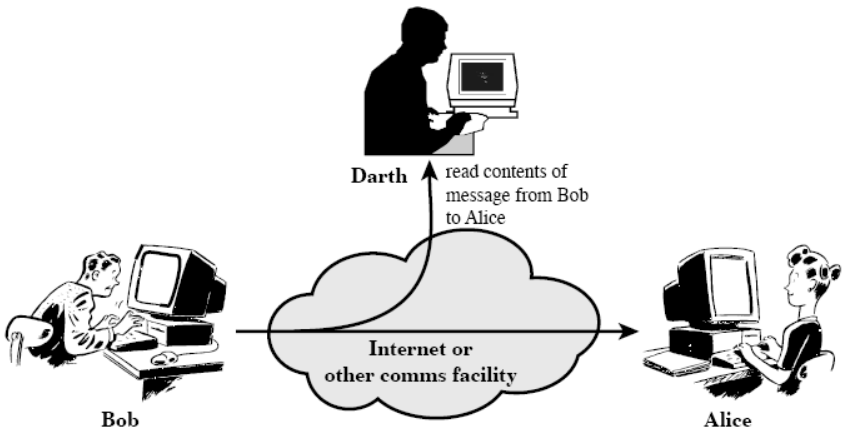
4.1. Interrelations



5. Attack categories, levels & causes

Passive Attacks - Eavesdropping

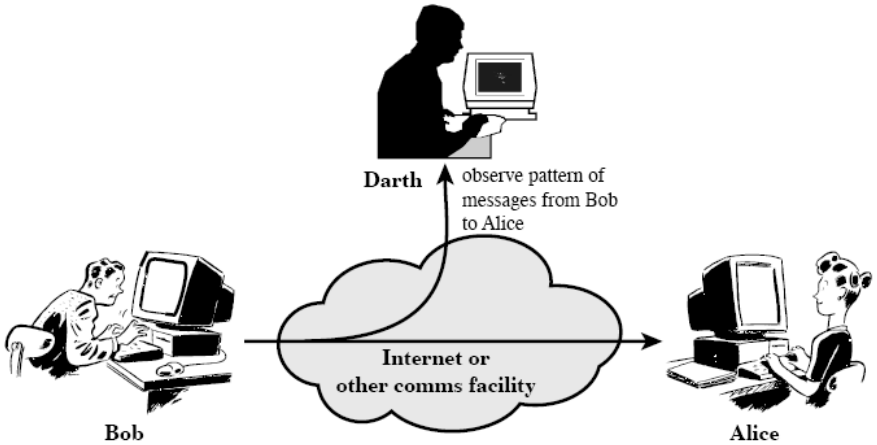
- Attacker intercepts the communication channel but does not actively intervene in the communication.



(a) Release of message contents

Passive Attacks – Traffic Analysis

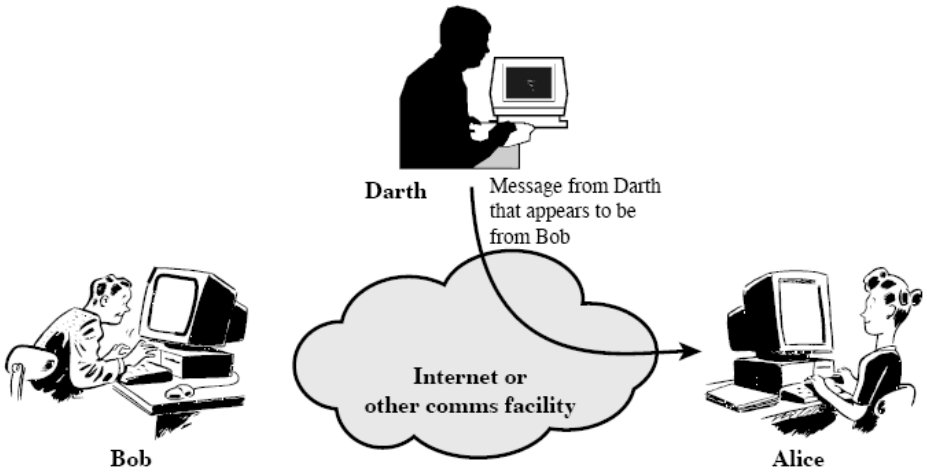
- With an encrypted data channel, a passive attacker may be able to perform a traffic analysis (who communicates with whom and when?).



(b) Traffic analysis

Active Attacks – Masquerade

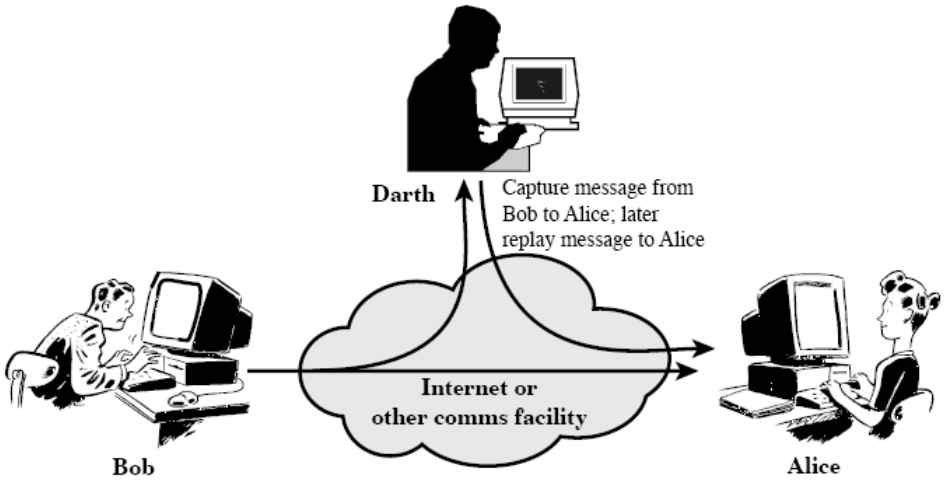
- Masquerade: Attacker impersonates someone else



(a) Masquerade

Active Attacks – Insertion & Replay

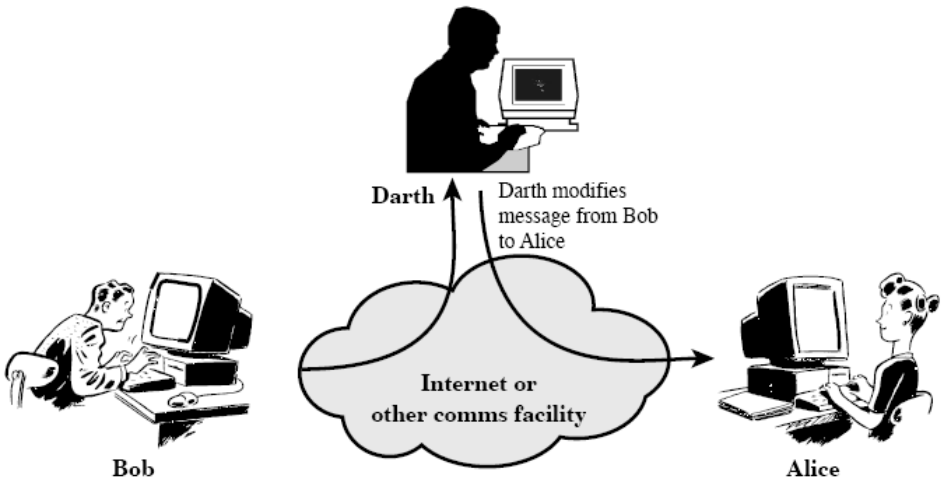
- Insertion: Attacker adds messages (parts) to a communication.
- Replay: Attacker sends recorded data again at a later time.



(b) Replay

Active Attacks – Modification

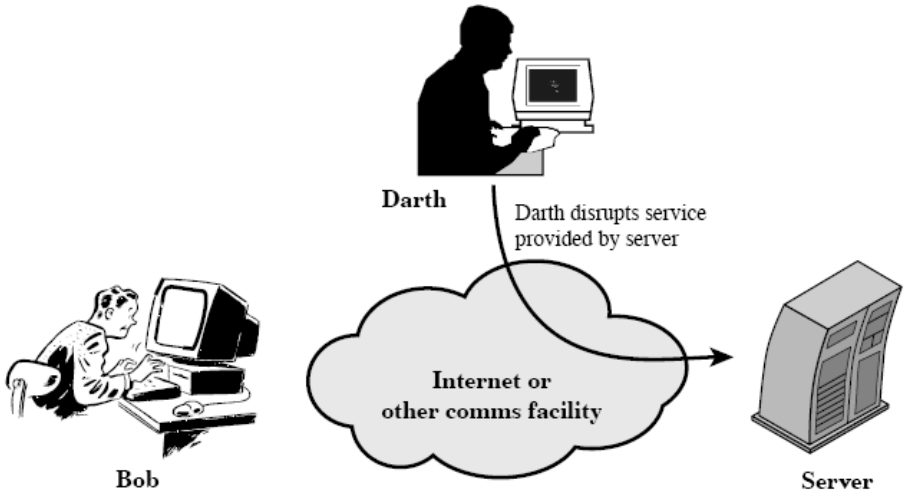
- An attacker changes a communication by delaying, changing or deleting messages.



(c) Modification of messages

Active Attacks – Denial of Service

- Attacker disrupts the availability of communication facilities



(d) Denial of service

5.1. Current attack levels (excerpt)

- Network
  - Botnets
  - (Distributed) Denial of Service, Reflection, Amplification
  - Spam (Unsolicited Commercial/Bulk E-Mail, Social Media Spam)
  - Man-in-the-middle attacks (e.g. for reading/modifying communication)

- Applications (especially web applications, see OWASP Top 10), e.g.
  - Injektion (z. B. SQL Injection, Command Injection)
  - Standortübergreifendes Scripting (XSS)
  - Betriebsübergreifende Antragsfälschung (CSRF)
  - Verunstaltungen
  - Pufferüberläufe

- User
  - Social Engineering
  - (Spear) Phishing
  - Scareware, Ransomware
  - Spam

5.2. Causes

- Missing/Deficient identity verification
  - e.g. weak authentication procedures, unilateral authentication, ...

- Missing/Deficient Input Validation
  - e.g. user entries in web applications are not checked on the server side

- Psychological deficiencies and obstacles
  - e.g. Social Engineering

- Factors cost and time in product development
  - e.g. short release cycles, missing tests, security not integrated into development processes

- Organisational deficiencies
  - e.g. missing security management, missing user awareness, ...

6. Attacker types & their motivation

- Amateurs (Script Kiddies)
  - Responsible for most (automated) attacks
  - Few in-depth knowledge Application of ready-made attack tools
  - Motives: Prestige, personal revenge, boredom

- Crackers vs. Hackers (Blackhats vs. Whitehats)
  - Crackers are malicious hackers (terminology unclear)
  - Deep technical understanding (students, computer scientists)
  - Motives: Prestige, intellectual challenge

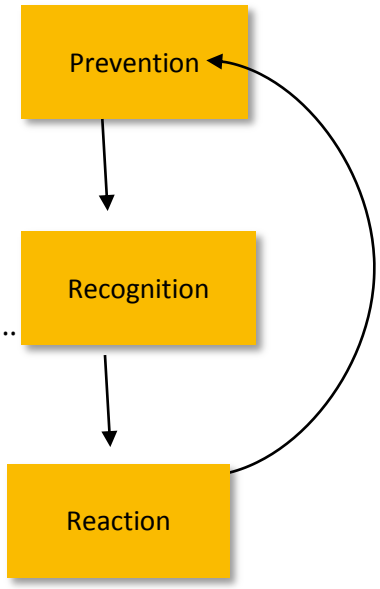
- Criminals (Cybercrime)
  - Classic criminals who change profession only for reasons of profit
  - Spamming, online extortion, bulletproof hosting/services, botnets (rental), …
  - Well organised networks with links to organised crime (e.g. Russian Business Network, Silk Road, …)
  - Motives: Profits

- Terrorists (cyber-terrorism)
  - IT as attack target Damage/destroy target infrastructure
  - Propaganda purposes
  - Communication and organization

- Countries and their military/ intelligence apparatuses (cyber war(fare), cyber espionage)
  - Switching off the IT infrastructure as a target of military attacks
  - Economic, political and military espionage (China, USA)
  - The Internet as a weapon
  - Examples: Stuxnet, Flame, Regin, NSA surveillance scandal

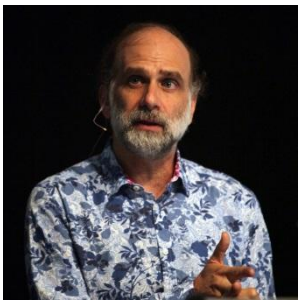
7. Classification of security measures

- Preventive measures (a priori)
  - e.g. authentication, access control, deployment of encryption, firewalls, hardening of systems, Security concept, security policy, creating awareness, ...
- Recognition measures
  - Dynamic at runtime
  - z. B. Firewalls, intrusion detection systems, log-analysis, ...
- Damage limitation measures (a posteriori)
  - e.g. tightening of controls, improvement of Security measures, Internet connection caps, ...



8. Continuous Holism in Information Security

Security is not a product; it's process!



Bruce Schneier
Photograph by Rama, Cc-by-sa-2.0-fr, from Wikimedia Commons

- Holistic approach is a critical success factor for IT/information security
  - e.g. firewall to filter traffic, but users can connect WLAN access points to the network
  - e.g. accesses to the systems are protected with passwords attached to the black board
  - e.g. users write down their Windows passwords on post-it's on their monitors
- IT/information security requires a combination of technical and organisational measures!
- IT/information security in the company must be supported by the management!
- IT/information security must be continuously lived and improved (examples?)!

8.1. Safety Standards

- Security management/security process, e.g.
  - ISO 27000 family
  - BSI basic protection standards 100-1, 100-2, 100-3 and 100-4
- System security (certification of products), e.g.
  - TCSEC (Trusted Computer System Evaluation Criteria)
  - ITSEC (Information Technology Security Evaluation Criteria)
  - Common Criteria for Information Technology Security Evaluations (ISO 15408)
- Action catalogs (technical/organizational), e.g.
  - BSI basic protection catalogues

- Other relevant standards ([IT] governance, compliance), e.g.
    - COBIT (Control Objectives for Information and Related Technology)
    - ITIL (IT Infrastructure Library)

8.2. Legal norms / laws

- Austrian Strategy for Cyber Security (ÖSCS)
    - Future: Austrian Cyber Security Act

- EU Directive on measures to ensure a high level of common network and information security (NIS Directive)
    - Incorporation into the Austrian Cyber Security Act

- EU Basic Data Protection Regulation

9. Literature

[Eckert2008] Eckert, C., Vorlesungsskript zur VO IT-Sicherheit, TU Darmstadt, SS 2008

[Stallings2006] Stallings, W., Cryptography and Network Security, 4th edition, Prentice Hall, 2006

[BSI1992] Bundesamt für Sicherheit in der Informationstechnik (BSI), IT-Sicherheitshandbuch – Handbuch für die sichere Anwendung der Informationstechnik, Version 1.0, März 1992,

[BSI2005] Bundesamt für Sicherheit in der Informationstechnik (BSI), IT-Grundschutzhandbuch, Stand 2005

NETWORKS

1. Basic terms

This part offers a comprehensive definition of basic terms which students may encounter in the area of computer networks. It is also necessary to say that suggested notes may not be considered as precise definitions; it rather refers to a complex area with many notable exceptions to the rule. All the same, students should have at least a broad conception of these keywords and should be able to efficiently use them.

Client means a computer connected to a network. It usually does not have any privileged status and his opportunities are similar to those of other clients. This term may also be encountered at labelling software which communicates with the server and, in this way; it provides a user with an essential service (FTP client, e-mail client etc.).

Server refers to a computer with a privileged status in a network. He is charged to provide the clients with essential services or functions, e.g. effective communication of clients, domain name transfer to the IP address (DNS), internet connection, print, sharing files etc. However, the server may be regarded as a client with respect to a different server. They are usually computers with a specific construction (special processor, disk arrays etc.).

Router is a device securing the packet routing. It involves a network element with a detailed knowledge of its environment and, simultaneously, it secures the packet routing in the right direction. Considering a packet as a letter, the router would play a role of postal service. Nevertheless, its application goes further; it guarantees high-quality services, secures TTL (a parameter limiting the lifetime period of packets so as not for the lost packet to stray forever); it should thoroughly explore its surroundings and find out network changes. At the same time, it calculates the best route to further hubs.

Packet includes a fixed dataset sent via network. Data are usually (almost never) not sent as an uninterrupted flow of information, but as datasets, i.e. packets. This elaborate system strengthens its security, secures right routing and effective use of the network capacity. Although the packet has a rigid structure, it usually contains sender's and receiver's address, information about the protocol, data or other supplementary information.

Network card is a hardware component (nowadays, usually integrated in the motherboard) which secures network connection. Contained information is transferred to packets and packets are collected to data. The card may carry out a specific protocol of data transfer (usually Ethernet) and also contains a port for cable connection (twisted pair, optic fibre, coaxial cable). Moreover, it contains a specific MAC address, which is unique all around the world and serves as the computer identification in the network, or alternatively, it generates a local address.

Port includes a number from 0 up to 65535, which identifies an application currently used as a means of communication. Individual applications are identified within TCP or UDP protocol. The ports enable an application servicing; secure (to a certain extent) the quality of services etc. As a matter of fact, there are fixed port numbers, e.g. SMTP contains 25, and POP3 110, HTTP 80 or FTP 21. All the same, however unusual is it, users or applications may agree on a different number of ports.

Medium represents a physical environment by which a network communication is carried out. It involves led media, i.e. classic cabling (optic cable, metallic cable, twisted pair) or wavelength media (air, vacuum), which use technologies such as Wi-Fi or GSM.

Protocol is a formal language defining the way of communication (IP, TCP, and IMAP4). It accurately describes the packet, i.e. the speed it should be sent; furthermore it defines security elements, corrects errors etc. Almost each layer of ISO-OSI model contains its own group of protocols. They are usually bound only to one layer.

Switch means an active network element connecting its parts. In contrast to a bridge, it allows connecting more than two network circuits. Smaller networks parts have a better access to the medium, as a result of which a higher transfer speed and network usage is secured.

Repeater refers to a simple device securing the signal transmission in a longer distance. It is situated within a certain distance from the transmitter and strengthens the incoming signal. In this way, it operates straight on the physical layer.

1.1. What is a computer network good for?

Companies install computer networks mostly in order to share resources and enable a direct communication. Resources include data, application and peripheral devices, e.g. external floppy disk drive, printer or modem. Direct communication involves sending messages, replying to messages or e-mail.

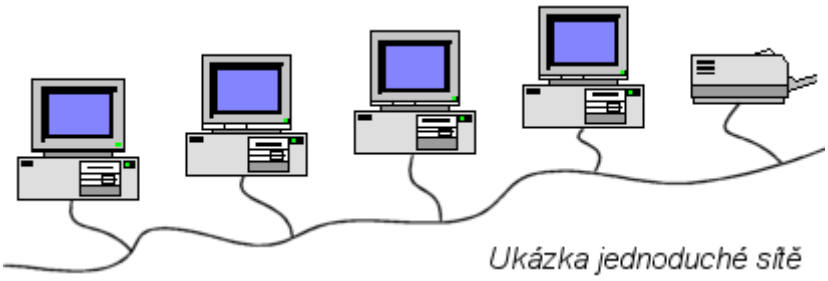


Illustration of a simple network 1

Computer network is a general term for technical devices by means of which a connection and data exchange between computers is carried out. In addition, it enables users to communicate according to the rules. The soundest reason for the network connection is sharing information and technical devices.

Picture 1: a star shaped network connection

According to the vastness, four computer networks may be distinguished:

- PAN (Personal Area Network) is a network covering a very small area (smaller than LAN area); network includes communication devices such as mobile phone or laptop,
- LAN (Local Area Network) is a local network covering a small area, i.e. an area larger than PAN, but smaller than MAN,
- MAN (Metropolitan Area Network) is a network larger than LAN and smaller than WAN,
- WAN (Wide Area Network) is a network covering a huge area, i.e. an area larger than MAN,
- VLAN means a virtual LAN (network). It is a virtual network created within LAN network which operates within one hardware component (cable). VLAN runs on the grounds of interpreting data, which are transmitted via the network. The data are interpreted according to the appropriate network.

2. Network topology

Network topology describes the physical connection. To put it more precisely, it is possible to discriminate between a logical and physical computer connection.

Ring topology refers to a circular connection of computers. In this way, one computer is always connected to two others. The biggest advantage of this model is that as long as one element falls out, the network may (supposed it is duplex) go on. Likewise, the nearest computers (in LAN network) usually communicate the most often.

Mesh is probably the least used topology. As a matter of fact, there are not any cardinal rules; thus, it is created ad hoc by progressively connecting computers. On the other hand, of interest may be its description, which is actually very simple. Logically, it is possible to say that Mesh is an internet considered as a whole.

Star represents one of the most common models of connecting; its individual stations

are connected to one central point (e.g. a server), which provides all the communication. A big advantage is its effective management and clear direction; on the other hand, a weak point is the server.

Tree refers to another frequently used topology. As a matter of fact, there is a complex structure of computers connected to a structure of a mathematical tree. In this way, individual computers may (unless end-to-end or root computers are to be dealt with) play the role of the server and client at the same time.

Bus concerns a model in which stations are progressively connected to one medium; thus, a concept of a shared medium is to be dealt with.

Naturally, there are other important topologies – linear, interconnection or an extreme case of a point-to-point connection. Usually, it is something in between.

2.1. Computer networks and their services

Computer network is a general term for technical devices by which a connection and data exchange is carried out. Furthermore, it enables users to communicate according to the rules. The main reason for the network connection is to share information and technical devices.

Picture: computer network

2.2. Network topologies

Topology refers to the way of connecting computers. In addition, it contains the information about the particular connecting computers and their communication. Possible methods of connecting are depicted below.

Bus topology Star topology Ring topology Unlimited topology

Picture: Topology – possible connecting computers networks

The connection of bus topology is provided by a transfer medium called bus, to which all end-to-end computers, i.e. network hubs, are connected.

Star topology refers to a star shaped network connection. It is the most frequent method of connecting computers based on a computer in the middle called a central computer. It is connected to other computers around. It is possible to imagine that one or more end-to-end computers are replaced with the central computer. Thus, the whole network is progressively growing.

Ring topology refers to a connection where one hub is connected to other two hubs in the way of a ring. However, this topology is not very fast since a huge number of hubs prevent fast data transferring to the target computer. Moreover, supposing one of the hubs does not work, the data transfer is not possible; thus, the whole network stops working. This sort of connection is not much used nowadays as the star topology is much more reliable. On the other hand, the costs of building up this network are lower.

3. What is the internet?

The internet is a global network similar to a common computer network. Computers are interconnected as a result of which they may communicate and share information. The internet consists in connecting already existing networks with a coherent structure and distribution.

The internet may also refer to computer system containing information and networks which provide us with the particular information.

Computers within the internet play the role of clients or servers. Servers provide internet services; subsequently, clients make use of these services. In addition, internet services secure data transfer to a client upon request.

3.1. Origin of the internet

During the Cold War, the US needed a functional management system able to connect the most important academic, government and strategic computers (countries, military bases etc.). The key requirement was to build up a strong network which would be running despite falling out of several network hubs and at least a partial communication would be secured.

As of 60s of 20th century, RAND Corporation Company was asked for solving the problem which lay in smooth running of computers even after a nuclear war. The main task was to devise a stable system despite a possible destruction of its parts.

In 1964, the same company came up with a comprehensive solution, which was based on two central principles.

- Network does not contain any central component
- Network is running although some of its parts are out of order

With all that said, the American army, i.e. Pentagon may be considered as the Internet creator. In 2010, 2 billion active internet users were registered.

3.2. The internet development

Zero phases

A zero phase began when the American army needed to establish an effective communication between government departments in case of nuclear war during the Cold War. The US Department of Defence, i.e. ARPA Agency (Advanced Research Projects Agency) set up ARPANET Network and managed financing of this project.

First phase

The first phase allowed access to the then most powerful computers, mainly to those of universities in the USA. In the late 1969, the first network hubs of ARPANET Network were placed at those universities. The main goal was to connect not only one computer, but the entire network. The development took place in all possible organizations; subsequently, local computer networks started to run. The beginning of 80s saw a rapid development of ARPANET which also went on in other networks. Thus, networks such as Usenet or BITnet were created. As a result, all the networks were connected to ARPANET. In 1987, more than 10,000 network hubs were registered; two years later, it was more than ten times more.

80s and 90s expanded services that people of today are familiar with and make a use of; it was mainly e-mail (1971), telnet (1972), TCP (1974) and its updates on TCP/IP (1978), DNS (1983) and its start in (1984).

In 1980, Pentagon decided that preferred protocols for the Defence Department would be TCP/IP protocols and 2 years later, all the computers connected to ARPANET network had to change for TPC/IP protocols. In this way, ARPANET became an incipient network and a conglomerate of co-existing and newly established networks, which was called the Internet, began.

Second phase

The second phase goes through the internet development in 1983-1992. This period is characterized by a dramatic growth of the internet and, mainly, its huge expansion beyond American continent. ARPANET also connected other networks such as NFSNET, EUNET, EARN, JUNET etc.

In November 1983, DNS domain system which allowed numeric addresses allocating domain names was introduced.

In 1989, WWW (World Wide Web), which afterwards became an integral part of the internet, was established.

The year 1990 saw decommission of ARPANET and its subsequent cancellation. As of then, NFSNET became the key internet network.

In 1991 the Czech Republic was connected to the internet. As of 1993, common users start to use it.

3.3. What is the internet good for?

Owing to the internet, its users may use a large number of services. These services are provided by computer programs and programs which communicate to one another by protocols. The basic internet services are as follows.

- **WWW** – a system of websites displayed via a browser; it uses http/HTTPS protocol.
- **E-mail** – electronic mail
- Sending messages via SMTP protocol
- Receiving messages via POP3 and IMAP protocol
- **IM – Instant messaging** – online, live communication ICQ, Jabber, Skype
- **FTP –** file transfer
- **DNS –** domain name system used for their better remembering
- Furthermore, plenty more protocols may be mentioned, e.g. file sharing (NFS), protocols used for connecting to a remote computer (telnet, SSH, VNC) etc.

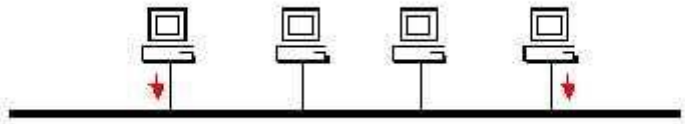
3.4. Ethernet

Ethernet is a technology for computer networks. It represents a set of technologies which deal with data transfer mainly in LAN networks. The most of its versions is subjected to IEEE institute standardization. Ethernet is implemented by the first (physical) and second (bond) layer of ISO/OSI reference model. Furthermore, Ethernet mostly uses a twisted pair or optical cable as a transfer medium; in the first versions, coaxial cables were dealt with. The next integral part is RJ_45 connector. There are several types of Ethernet depending on the type of cabling and transfer speed; the transfer speed was from 10 Mbit/s up to 100 Gbit/s.

Nowadays, Ethernet has become the most frequent network technology. This technology is based on a very simple principle called CSMA/CD irrespective of it being a common 10 Megabit Ethernet, or its faster mutation (Fast and Gigabit Ethernet).

A well-structured cable system may use various kinds of network technologies based on different transfer methods, e.g. Ethernet, Token Ring, CDDI, ATM etc.

CSMA (Carrier Sense Multiple Access) is a station ready to transmit data and "checks" whether the transfer medium (cable) is used by another station. In such a case, the station tries to access it later until the medium is free. The moment the medium is free, the station starts to transmit its data.



CD (Collision Detection) refers to a station which, during transmission, observes whether

the medium shows a signal corresponding to transmit levels (i.e. so as not to transmit signal 1 the moment it transmits signal 0). Such a case of a complex signal interaction is called **collision**. In case of **detection**, the station collision sends out JAM signal and both (all) stations at the same moment of transmission generate a random time value after which they will try to repeat the transmission.

As a matter of fact, such effectiveness brought down prices of AC adapters and active elements and, consequently, Ethernet considerably expanded. All the same, the effectiveness of such a solution brought one grave disadvantage; with an increasing number of network hubs, simultaneously, a number of collisions increase; therefore a theoretical network throughput decreases. The set of hubs of which the common activity may cause a collision is called **collision domain**. Logically may be deduced that the collision domain should be as small as possible. Applied active elements have a different relation to the collision domain; on the one hand, some extend the collision domain and, on the other, some separate it. Therefore, by carefully choosing them, the network throughput may be controlled.

In addition, apart from the collision domain, a term **broadcast domain** has been applied. The computer network contains two kinds of packets; these are unicasts and non-unicasts. Unicasts refer to packets with a particular addressee provided by the network address. On the other hand, non-unicasts tend to use a group address and are either for all network users (broadcasts), or selected group of users (multicasts). The main issue is that a computer has to occupy itself with non-unicast even though it is not meant for it. Therefore, with an increasing number of network hubs, simultaneously, a large number of non-unicasts increases in the broadcast domain. For that reason, it is necessary to keep the broadcast domain in the reasonable size. Applied active elements have a different relation to the broadcast domain; consequently, by carefully choosing them, the network throughput may be controlled.

Packet format

As has been said before, all Ethernet speed modifications follow the same communication method CSMA/CD. However, they also use the same packet size and format. Ethernet packet is defined on the 1st and 2nd OSI layer.

4. Standardization of computer networks, TCP/IP stack

4.1. Standardization of computer networks

The general strategy of leading producers involved creating and maintaining such specifications that provided users with compatibility strictly with products of their company.

These computer networks, labelled as homogenous or closed systems, had a major deficiency in adapting to both, the manufacturer of technical equipment and the manufacturer of software (here are some of the proprietary protocols – DECnet from DEC Company, SNA – Systems Network Architecture from IBM Company, SPX/IPX – Sequenced/Internet Packet Exchange from Novell Company etc.). In contrast to these closed systems, open or heterogeneous systems refer to computer systems which were devised according to the established international standards and may be purchased from more independent producers. The application of open systems allows programs operating in local or remote network systems to closely cooperate and enables a unified communication of users of these applications.

The term OSI (Open Systems Interconnection) concerns technical equipment and computer network software. In the area of computer networks, international organizations CCITT (Comité Consultatif International de Télégraphique et Téléphonique is one of the three central committees ITU – International Telecommunications Union) and ISO (International Organization for Standardization) are the most important ones and deal with establishing adequate standards. As a matter of fact, government institutions or organizations responsible for the telecommunication in individual member countries are key members of CCITT. CCITT publishes detailed recommendations which, after a general approval from ITU, become international standards. In contrast to this organization, ISO is a voluntary non-governmental organization, whose members are standardization institutions. Other important organizations are IEEE (Institute of Electrical and Electronics Engineers), which is a prestigious American association of electrical and electronic engineers. This association issues periodicals, holds conferences and creates a professional body which establishes professional standards. IEEE 802 group of standards is notable within the area of local computer networks. In 1979, ISO organization adopted a standard labelled Reference Model of Open Systems Interconnection (RM OSI).

Network communication

The main network activity involves data transmission from one computer to another. This complex problem may be divided into these tasks.

- examining data
- dividing them into usable data banks
- supplying each data bank with useful information (source and goal)
- providing relevant information about timing and error detection
- transferring data to a network and sending them to a proper place

4.2. Basic terms RM OSI

Layer is accurately defined by its useful functions. Each layer, except for the highest and lowest, interfaces the layer directly below and above. The highest layer interfaces the application process. In addition, the lowest layer interfaces the physical media.

Entity is an object effectively operating within the particular layer. The term usually refers to a program group. Lower levels contain hardware (I/O port). Actually, the direct supplier and service user do not refer to a layer, but a particular layer entity.

Protocol includes entities in the identical layers of various open systems; they **communicate**

Service refers to entities of a particular layer performing its function; furthermore, they provide the higher layer with adequate services. In order to carry out these functions, they use services of the layer directly below. Individual services are offered by so called Service Access Points (SAP), which are identified via their addresses. These services may be divided as follows:

- **Connection-oriented Services** - two entities on the same levels must at first establish a connection before starting a direct communication. After the successful connection, the sender sends a message and the **receiver**, on the other hand, receives it. As a matter of fact, this kind of service is similar to the phone connection.
- **Connectionless Services** does not rely on establishing a strong connection between a **sender** and receiver. **Instead**, they consider individual parts of transferred data as single wholes supplied with the address of their final receiver and are delivered irrespective of other messages. Individual messages may be transferred in different ways.
- **Reliable Service** provides services that never lose any data. Usually, it involves services maintained by an effective mechanism for confirming messages.
- **Unreliable Service** does not provide confirming messages, but offers services of a high reliability.

4.3. ISO/OSI Reference Model

This popular model was designed by ISO international organization as a unified stand-

ard for interconnecting systems of various kinds and conceptions defined by different producers. The produced model contains seven layers. These seven layers create a complex hierarchy beginning with applications on the top and ending by physical connections at the bottom. OSI reference model includes two models of communication.

- horizontal – a protocol-based model by which programs and processes of different computers communicate,
- vertical – a service-based model by which layers of a single computer communicate,

As a matter of fact, an effective communication requires following elements; it requires at least two sides willing to communicate; therefore, a common language (protocol), by means of which the sides will communicate, must exist. In addition, vertical layers communicate via API (Application Program Interface).

Application layer, presentation layer, session layer, transport layer, network layer, data link layer, physical layer.

4.3.1. APPLICATION LAYER

This layer precisely specifies the environment in which network applications communicate with the network services. The end user uses the networks for running the applications, file transfer, mail, remote login etc. Application layer protocols mainly contain application programs; also network superstructures, which enable the station to connect to the network, may be found there. Programs and protocols providing application layer services are as follows:

- NICE (Network Information and Control Exchange), which secures monitoring and network administration.
- FTAM (File Transfer Access and Management), which is responsible for remote file administration.
- X.400, which specifies protocols and functions for forwarding messages and electronic

mail.

- CMIP, which is responsible for network administration based on the framework formulated by OSI.
- Telnet, which provides a terminal emulation and remote connection.
- Rlogin, which provides UNIX environment with a remote connection.

4.3.2. PRESENTATION LAYER

This layer manages formatting of data transfers. The data, which are transferred via the network, may be texts, digits or general data structures. Furthermore, it contains specifications for coding and decoding charsets; a data compression or their ciphering might be also carried out within this layer. In addition, presentation layer provides services to an application layer situated directly above and uses a session layer below.

4.3.3. SESSION LAYER

This layer stimulates processes which control data transfers, detects transmission and transfer errors and files records on broadcast transmissions. In addition, the layer keeps and disturbs sessions between end-to-end participants (the user and target computer). In regard to starting the session, the layer requires establishing a connection on the transport layer by means of which a communication between both session participants is carried out. Likewise, it decides who may transmit signals or, alternatively, it disturbs the existing session. Session layer protocols are as follows:

- ADSP (AppleTalk Data Stream Protocol) allows two network hubs to make a tenuous connection for a data transfer.
- NetBEUI refers to an implementation and development of NetBIOS.
- NetBIOS uses 5th, 6th and 7th layer and offers session monitoring services.
- PAP (Printer Access Protocol) allows the access to PostScript printer in AppleTalk network).

4.3.4. TRANSPORT LAYER

This layer allows functioning of appropriate connection establishment, disintegrates data to smaller parts, i.e. packets, to which the transferred data are gathered; upon receiving, the data are taken out of the packet and folded into the original shape. Therefore, it secures transferring of randomly big messages despite individual packets having a limited size. As a matter of fact, the transport layer is fairly essential since it is situated between upper and lower layers, which are network-oriented.

4.3.5. NETWORK LAYER

This layer describes processes which route data between network addresses and check whether the complete messages have been sent on time. In addition, it secures a convenient routing of packets. Furthermore, the network layer must be aware of the particular network topology, i.e. the way of the direct interconnection of individual network hubs.

4.3.6. DATA LINK LAYER

This layer, also called a bond layer, deals with processes which detect and correct errors on the data level during the data transfer between the physical layer and layers above the physical layer. Data link layer creates packets of relevant network architecture. All the same, the transfer route is inclined to errors as a result of which received bites are different from those that have been sent. Physical layer does not deal with the importance of individual bites; such sort of errors is detected to the data link layer; it checks whether the whole packets (sending different kinds of checksums etc.) were transferred correctly. Furthermore, the data link layer provides the sender with a direct confirmation of receipt of flawlessly transferred packets, while an error requires their repeated transmission.

4.3.7. PHYSICAL LAYER

This layer imposes electric, mechanical and functional requirements for network data processing. By and large, the task seems to be easy; it secures transfer of individual bits between a receiver and sender.

4.4. TCP/IP Model

TCP/IP stands for Transmission Control Protocol/Internet Protocol; it refers to a standard file of protocols used e.g. in the Internet Network, which has become the largest network in the world.

History points out that in 1969, DARPA (Defence Advanced Project Agency) launched a research and development project aimed at creating an experimental network with packet switching. This network was called ARPANET and was designed to study techniques allowing reliable and supplier independent data communications. On the whole, this experiment was highly successful and a large number of organizations started to use its services for a common daily communication. In 1975, ARPANET changed from an experimental network to an operating network. The basics of TCP/IP protocol were laid down when ARPANET had already been an operating network. This protocol was accepted as a Military Standard in 1983 and, at about this time, the term Internet spread all

over the world.

Advantages of TCP/IP

- Open protocol standard freely available and developed irrespective of a particular technical equipment or operating system.
- Complete independence of a particular physical network hardware as a result of which it is possible to run TCP/IP within a large number of networks. Therefore, TCP/IP may run on Ethernet, token ring, phone lines and practically on each physical transfer medium.
- General addressed scheme allowing any TCP/IP device to directly address any other device in the entire network.
- Standardized high level protocols providing widely available user services.

4.5. TCP/IP Protocol architecture

Protocols draw up a formal code of conduct within data communications. In homogeneous networks, the set of rules is formulated by the system supplier. TCP/IP creates a heterogeneous network with open protocols which do not depend on differences between operating systems and architectures; they are available to everyone and their development and modifications are subject to the terms of the contract. The most precise information on TCP/IP is provided in RFC (Request for Comments); these documents contain the last updates of all standard specifications.

TCP/IP is usually considered as a model composed of fewer layers than OSI model. Most of the TCP/IP descriptions define the protocol architecture by three to five functional levels.

Network Interface Layer

This layer deals with everything associated with controlling a particular transmission path, direct transmission and reception of data packets. Moreover, it depends on a particular transmission technology (i.e. all industry standards – Ethernet, IEEE 802.x, and FDDI).

Internet Layer

This layer is sometimes called IP layer, in regard to IP protocol, and secures for individual packets to be sent from the sender to their actual receiver irrespective of there being a direct connection between them. With respect to the unconnected character of the transfer (IP protocol), a basic datagram service is provided on the level of this layer.

- Internet protocol is a cornerstone for the internet. Its functions include:
- Datagram definition, which is the basic transmitting unit,
- Definition of the internet addressing scheme,
- Data transfer between the network and transport layer,
- Directing datagrams to distant destinations
- Providing fragmentation and datagram composition

Datagram refers to a packet format (packet is a data block, which contains information

necessary for its delivering) defined by the internet protocol.

Transport layer

This layer is also called TCP layer according to the protocol. The layer is responsible for securing transmission between end-to-end participants, which, in case of TCP/IP architecture, are represented by application programs. According to their requirements, the transport layer may regulate the data flow in both directions, secure the transmission reliability and also change the unconnected character of the transfer to the connected one within the network layer.

Application Layer

Application layer is the very up layer of TCP/IP architecture; its entities represent individual application programs, which, in contrast to RM OSI, communicate directly with the transport layer. Possible presentation and relational services are provided by application programs themselves.

4.6. Architecture of communicating systems

Definition of terms

- **System**
  an independent whole able to stimulate processes and transfer information
- **Network architecture**
  system of layers, services, functions and protocols
  correspond to the structure of network equipment
- **Open architecture**
  adequate standards describing the architecture is publicly accessible
  all systems meeting the standards are inter-connectable
- **Layer protocol**
  rules of cooperation of entities on the same layer and other systems
- **Interface protocol**
  SW interface
  rules of cooperation of adjacent layers
  service primitives are used
  communication via Service Access Points (SAP)

5. ISO/OSI Model, selected protocols

5.1. Computer networks – ISO/OSI Model

ISO/OSI model refers to a communication model labelled 'International Standards Organization / Open System Interconnection'. It concerns a recommended model defined by ISO organization in 1983, which divides a common communication between computers into seven interconnected layers. The concerned layers are referred to as a Set of Protocol Layers.

The main task of each layer is to provide the following upper layer with particular services and not to burden the upper layer with details about the particular service provision. Before transferred from one layer to another, data are divided into packets. Subsequently, in each layer, all packets are supplied with additional information (formatting, address) necessary for a successful transmission over the network.

The suggested model shows the following layers (each upper layer uses functions of the lower one).

A brief description of individual layers – the individual layers appear in the same order as mentioned above

1. Physical layer

It defines means of communication with a portable medium and technical means of the interface. Furthermore, it defines physical, electric, mechanical and functional parameters concerning physical interconnection of individual components; i.e., it deals with hardware.

2. Data link layer

It maintains the integrity of the data flow from one network hub to another. This activity involves data block synchronization and their flow management; i.e. it also deals with hardware.

3. Network layer

It defines protocols for data directions, by means of which the information transfer to the required target network hub is secured. As a matter of fact, the appropriate direc-

tion, unless used, does not have to be in the local network. It also refers to hardware; however, when the PC deals with the direction between two network cards, it concerns software.

4. Transport layer

It defines protocols for structured messages and secures a flawless transfer (carries out some of the error checks). Moreover, it secures a partition file into packets and confirmation. It deals with software.

5. Session layer

It coordinates the communication and maintains the session as long as needed. Furthermore, it performs security, login and administrative functions. It concerns software.

6. Presentation function

It explores the way of data formatting, presentation, transformation and coding; i.e. it deals with diacritics, punctuation, compression and decompression and data encryption. It concerns software.

7. Application layer

It represents the uppermost layer. It explores the way of communication of a network with applications, e.g. database systems, electronic mail or programs for terminal emulations. It uses services of lower layers as a result of which it is isolated from problems of technical networking resources. It is software-based.

5.2. ISO/OSI Reference model – seven layers

Designers of ISO/OSI reference model concluded that an optimum number of layers of network software are seven. However, which layers and tasks are to be dealt with? Let's proceed from the lowest to the uppermost.

1. Physical Layer

Its main task seems to be very simple – to make a transfer of individual bits between the receiver and sender via the physical transfer path, which is under a full control of this layer. Nevertheless, a lot of issues of technical character must be dealt with – e.g. the voltage level of logic 1 and logic 0; how long one bit "lasts"; how many contacts and which shape cable connectors should take on; which signals are transmitted via these cables; of which importance they are; their time course etc. Therefore, these issues rather belong to the area of electro-engineers and technicians.

2. Data Link Layer

Physical layer usually provides means of individual bit transferring as its standard services. Using these services, the right above data link layer (sometimes called **bond layer**) must secure a flawless transfer of data blocks (in the scope of hundreds of bits) labelled as **frames**. Since the physical layer does not interpret individual transferred bits, the da-

ta link layer must correctly define the beginning and end of the frame and its individual parts.

In fact, a various disturbances and failures may occur on the transfer path as a result of which received bit values differ from those that have been sent. Since the physical layer does not deal with individual bits, these errors are detected up to the data link layer. This layer checks whole frames whether they were correctly transferred (according to the checksum; see 3rd part of our series). Furthermore, it provides the sender with information about a flawless frame transfer while, in case of damaged frames, it requires their resending.

3. Network Layer

Data link layer secures transferring of whole frames; however, only between two network hubs, where a direct connection is established. Nevertheless, what to do supposing the connection between a receiver and sender is not direct, but it goes through one or more intervening hubs? As a matter of fact, network layer must take an action and secure a convenient routing of transferred frames – labelled as packets. The network layer chooses the correct path (or route) via intervening hubs and, also, secures progressive transferring of individual packets from the original sender to the end receiver on the route.

Therefore, the network layer must "consider" a particular network topology (i.e. the way of direct interconnecting of individual hubs).

4. Transport Layer

Network layer provides the right above layer with services securing packet transfer between two random network hubs. As a result, the transport layer is screened out of the actual network topology and creates illusion of every network hub having a direct connection to any other network hub.

Consequently, the transport layer only secures an end-to-end communication; i.e. a communication between the original sender and end receiver.

Moreover, the transport layer secures building up individual packets upon the data transfer into which the transferred data are divided; analogically, the data are taken out of the packets and built up into their original shape upon receiving. In this way, a transfer of any large messages is secured despite the individual packets having a limited size.

5. Session Layer

This layer establishes, maintains and breaks off **sessions** between end-to-end participants. While establishing a session, this layer requires a connection to the transport layer through which a communication between both session participants is established. This layer is responsible for an occasional communication management (i.e. outlining transmission timetable when two participants simultaneously occur). In addition, the layer is responsible for everything needed to close the session and break off the actual connection.

6. Presentation Layer

The data transferred via the network may be, apart from others, textual, numeric or general data structures. All the same, individual hub computers may use different internal representation of these data; i.e. central computers of IBM Company use EBCDIC character code while most of the others work with ASCII code. Likewise, one computer may display integers in the supplementary code while another one may work in a direct code etc. – this layer is responsible for required conversions of transferred data.

The layer also carries out potential compressions of transferred data, alternatively their ciphering.

7. Application Layer

End-to-end users make use of computer networks via various network applications – electronic mail systems, file transfer, remote login etc. As a matter of fact, incorporating all the different applications directly into the application layer would not be relevant due to their enormous variety. For that reason, the application layer involves only a part of these applications which provide common, i.e. generally applicable mechanisms. For example, considering electronical mail, the particular part which secures network messaging is also an essential part of the application layer. For all hub computers which use the same electronic mail system, this part is identical. The user interface of the electronic mail system, i.e. the particular part which is actually employed by the user and which enables a user to read messages, replies to them, create new ones and submits them for sending, is no longer considered as a part of the application layer since it may significantly differ in each particular hub; e.g. controlling (by means of line commands, various kinds of menu, with or without windows etc.). Another typical example may be a terminal emulation needed for remote login. On the whole, there are a large number of various terminals and to make a required adaptation between two random kinds of terminals is quite impossible. As a consequence, only one 'reference' terminal – so called virtual terminal – is implemented. Each particular type of terminal contains only one adaptation between this virtual terminal and the actual terminal. Nevertheless, the means for working with the virtual terminal are included in the application layer (since they are the same) while means for its adapting to the particular terminal are not involved in the application layer.

6. Wireless communication technologies

Wireless networks; principles, standards, components, communication mode, application and properties,

Standards

A development of wireless networks took place similarly to cable networks; initially spontaneously, then it was necessary to define standards which would ensure a common network cooperation. Main producers of the wireless technology formed WECA Alliance (Wireless Ethernet Compatibility Alliance), which imposed requirements for its effective management and in this way ensured a common compatibility. Upon complying with the conditions, the product is awarded Wi-Fi certificate, which confirms the compatibility with products from other producers. The wireless standard itself was based on Ethernet; therefore they have similar features; i.e. CSMA/CD access method and a similar packet composition. There are several standards for LAN wireless networks, of which the properties are suggested in the table. 801.11g standard is backward compatible with the older and slower 802.11b (they may cooperate; however only on a lower speed). 802.11i standard was based on 802.11g standard, the former of which uses a safer authentication and ciphering algorithm.

Components

Wireless components communicate in two useful ways:

- Ad hoc, this refers to a direct connection of several computers – from two to five. Each computer communicates with another one on the equal level; i.e. they are equal (the organization is similar to a peer to peer network). The main advantage is a quick installation and a very low price (except for client network adapters, other hardware is not required). It allows sharing files and the Internet, print over the network and other things commonly occurring in LAN networks. On the other hand, the key disadvantage is that all connected devices must be in range; i.e. everyone has to see everyone. The next shortcoming is a ridiculously easily-established connection as a result of which its security may be compromised.
- Infrastructure mode is based on Access Point (AP). It functions as a server through which all data flow between network clients (the organization is similar to client/server network). The key advantage of its application is a possibility of filtering and supervising the operation including making networks available to different clients. Thus, the elimination of unconscious attempts to establish ad hoc connection is secured; the entire flow must be directed to AP, which provides an adequate network protection. The access point is definitely not needed supposing a wireless connection is established only occasionally and only between several devices. However, provided a small home network is built up or shared in a household or small office, an access point is usually required (tighter network security).

Access Point

AP is a base of a wireless network. It establishes a connection between wireless end-to-

end points and a server by means of placing it within a metallic LAN network. For that reason, AP contains a radio part – transmitter/receiver and a cable part – RJ-45 socket for a twisted pair connection. AP involves hubs transmitting the signal. A lot of producers offer powering of AP by a twisted pair, through which the point is connected to the fixed network. Thus, the access point (in a hardly accessible place) does not have to bear two cable lines. The access point and its counterparts – client adapters work only unless there is an obstacle – between AP and computers placed in the wireless network must be clearly visible. As a consequence, access points may be found in the uppermost parts of rooms. Their positioning must consider possible sources of radio signal interference; i.e. metal constructions (even in walls), electrical interferences (e.g. microwave ovens operate in the range of 2.4 GHz, wireless phones, wireless speakers etc.). As long as a direct wireless interconnection is required, Wireless Bridge (WB) may be applied – access points of a bridge function, filtering packets between networks.

Client adapter

It refers to a unit through which a PC is connected to an access point. Basically, it refers to a network card (with an aerial). It is designed for PCI and USB slots. Laptops may apply a wireless network card according to PC Card standard; however, a lot of laptops run on an already integrated wireless network interface (which facilitates and marks down a wireless network set-up).

6.1. Wireless network properties

6.1.1. SPEED

In contrast to metallic networks, its speed is considerably lower. As a matter of fact, theoretically maximum attainable speeds are suggested in the table, which also provides adequate standards. Actually, these standards are hard to achieve, the fact of which is caused by a weaker accessibility of the signal between radio stations, which results in a skip to a lower transmission speed.

On the whole, it means that a longer distance (e.g. over 20 metres) or an obstacle (e.g. a quality metal frame) dramatically decreases the speed by half or quarter. As long as the group of receivers is connected to the same access point, and thus they are physically located within one network segment, the capacity must be divided by an adapter. As a result, a lot of collisions occur (CSMA/CD character). The next variable bringing down the feasible speed is a careful management of protocols within upper layers as a result of which the actual bandwidth for pure data considerably decreases again. All the same, under favourable conditions, the maximum bandwidth for a useful data load is about a half of the nominal values; i.e. 5 Mb/s B line and 25 Mb/s G line.

6.1.2. SECURITY

SSID (Service Set ID)

It refers to an access point which is visible to all the clients who are located within its range. For that reason, SSID serves as a logical indicator of a particular wireless network. It may be manually fixed to the station, or the access point regularly transmits the information about SSID, or SSID transmission may be switched off in the situation of which the client himself asks for SSID.

WEP (Wired Equivalent Privacy)

It refers to an old security system of wireless networks according to original IEEE 802.11. Standard. The aim of WEP was to provide wired computer networks with a top security (e.g. twisted pair) since the radio signal may be easily picked up even within a long distance without a physical contact with the computer network. However, WEP was broken through in August 2001; therefore, it should be replaced with WPA2 according to IEEE 802.11i standard.

WPA (Wi-Fi Protected Access)

This refers to a security system of wireless networks. In order to break WEP security system through in 2001, Wi-Fi Alliance improved WPA security system as a part of the then prepared standard IEEE 802.11.i in 2002. In addition, Wi-Fi Alliance holds the rights for Wi-Fi and WPA trademark and certifies their products.

WPA2

It implements all necessary components of IEEE 802.11i. It uses Advanced Encryption Standard (AES) block cipher while former WEP and WPA use the current cipher RC4. 802.11i; the architecture contains the following components: IEEE 802.1X for authenticating (i.e. Extensible Authentication Protocol – EAP).

Wireless PAN

WPAN connects individual devices in a relatively small area, which is very easy to access for a person connected to this network. For example, headphones may be connected to a laptop by Bluetooth or infrared light; in this way, a small private wireless network may be created (WPAN). Zig Bee also supports WPAN applications. As a matter of fact, private Wi-Fi networks became a commonplace for an ordinary consumer to the extent of the integrated equipment within a large scale of electronic devices. In addition, devices such as 'My Wi-Fi' from Intel and 'Virtual Wi-Fi' from Windows 7 facilitate setting and configuration of a private wireless network.

Wireless WWAN

A large wireless network WWAN refers to a wireless network covering large areas. These networks may be used for connecting office branches or as a public access system. Wireless connection between access points is usually carried out by a point-to-point microwave line using a parabolic reflector of 2.4 GHz frequency. A typical system runs on entrance gates of basic stations, access points and wireless signal bridge. The other con-

figuration consists of network systems where each access point passes the signal.

Wireless MAN

Wireless metropolitan networks WMAN refer to wireless networks which are connected to several local networks. WiMAX is a type of MAN wireless network and is defined by IEEE 802.16 standard.

6.1.3. APPLICATION

Its application includes mobile phones, which are an integral part of the everyday wireless communication and facilitate the communication, intercontinental network system, which employs satellites for communication all over the world. Common people and salesmen use wireless networks for fast sending and sharing data irrespective of them being in a small office, or anywhere in the world

7. Virtual networks (VLAN).

VLAN concerns a Virtual Local Area Network. It refers to a functional LAN network within which another one, a virtual network, is created; it runs on only one hardware device (physical cabling). On the whole, VLAN operates on an individual pre-setting of already active network components of a new virtual mode and model.

VLAN works on a principle of tagging data transferred via the network. The data are tagged according to the appropriate virtual network. The key goal is to properly interconnect more local LAN on the level of the second network layer of ISO/OSI model. Moreover, a great advantage is a good configurability.

Regarding a high-quality network security, VLAN easily discerns users of individual end-to-end stations and ever-running applications (e.g. Linux Demon) independently on the active network device.

In this way, unauthorized access of a non-ethical hacker to a computer network is entirely eliminated. Thus, the most common kind of a cyber-attack aimed at computer networks – DDoS may be resisted in the very beginning.

7.1. What is LAN network good for?

First launches of VLAN technology took place in the middle of 90s of the last century. One of the main reasons was to form a group of particular users which will successfully communicate and access their files and information in an easier way. The next reason was stagnation in the development of Ethernet technology.

Of importance is also the fact that VLAN technology easily discerns a network communication which has recently shared a network on an appropriate network device – switch.

As a matter of fact, using a switch port offers the most effective and practical solution.

VLAN technology developed in 1995; however, it was based only on a close source technology; yet, it has been only a couple years ago since it developed mainly to middle-sized and large companies although an adequate standard has already been defined.

Main reasons of VLAN creation:

- *dividing network users* into groups, sections or by particular services instead of a physical position and communication sections between these groups
- *reducing network broadcasts* which became a problem a couple years ago
- *reducing the collision domain* in time, when hubs were used instead of switches

The idea of a logical group of users, which is mentioned in many materials, and therefore a creation of VLAN observes:

- *Organizational structure* – provided most of the communication is carried out within the section with printers, file servers etc.; provided there is no communica-

tion between individual sections; only a few services (mail) are common.

- *Services* – VLAN associates workers who use the same services (accounting, DB etc.).

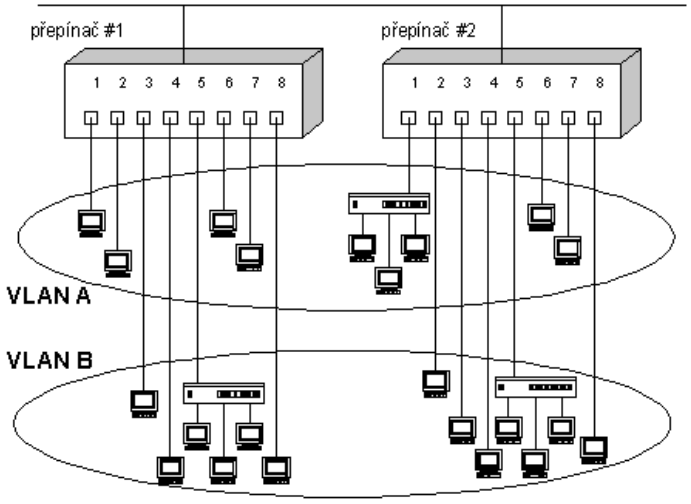
Major advantages of VLAN

Since VLAN membership may be defined in different ways, VLAN are typologically divided into networks with port-based and policy-based memberships. A more distinct division distinguishes four types of VLAN membership:

- ports;
- MAC addresses of hubs;
- network protocol or network addresses of hubs;
- Group of IP broadcasting.

Port-based VLAN

This historically first type of virtual networks defines a network membership for individual switch ports (group of ports). The first VLAN implementations did not allow expanding of virtual networks to more switches; they focused only on one switch. Yet, the next generation did not allow it; thus defined network is depicted in the picture below.



Picture – a port-based VLAN membership

Grouping ports is the principal method of creating virtual networks. Although very clear and simple, its basic limits consist in redefining a membership in relation to any moving of the user station between individual switch ports (i.e. such changes which would cause a change in the membership in the virtual network).

MAC address-based VLAN

MAC address is "hard-pressed" in the network adapter circuit; therefore, thus defined virtual networks may be considered as user-based VLAN. Supposing a user changes his connection (relocates his station to another switch port), his VLAN membership remains the same.

This method involves a rigid manual definition of a membership for all networks of the station. Another huge disadvantage is a risk of a substantial power reduction in case of a shared segment with stations in different VLAN being connected to the switch port. The next, rather a minor, problem may be a situation in which users, and their laptops, change their position and connect by a steadily-connected docking stations. Thus, a definition MAC address changes with a location (network adapter is mostly a part of the dock). Although it poses only a minor problem, certain VLAN limits, which are imposed by MAC address-based membership, are clearly illustrated.

However, the possibilities of user redefining of own MAC address directly in the operation systems adds a further complication, e.g. in regard to the security.

Protocol or address-based VLAN

Thus defined virtual networks are based on the information from the third, OSI-based network layer. In multiprotocol networks, hubs may be divided into individual VLAN according to operating network protocols, or e.g. in networks with TCP/IP protocol according to the subnetwork address.

Although information about the network layer is concerned here, it is important to realize that its utilization for routing is not dealt with. Despite the fact that the switch must examine the packet to specify the IP address and in this way VLAN membership, no routing calculations are carried out. The switch does not use routing protocols (such as RIP, OSPF); as a consequence, VLAN defined according to the information from the third, network layer, must be regarded as a network with a flat topology interconnected by switches or bridges.

Switching even in the third, network, layer and implementation of switches with a built-in router capability slightly alleviate the problem at the first glance. However, it refers to different functions – on the one hand, merely determining VLAN membership based on a network address; on the other, a full use of router capabilities based on routing protocols and calculations. It is also necessary to say that the communication between individual VLAN requires using routers – regardless of it being classic ones, or built-in in the switches with routing functions.

All the same, defining network layer-based VLAN has a lot of advantages, e.g. users' mobility or, to put in more precisely, their stations without a necessity of reconfiguring VLAN membership, a possibility of forming groups which provide particular services or applying and eliminating the need of packet tagging containing the information about VLAN membership when communicating with each other (see the following).

Group broadcasting-based VLAN

Group broadcasting in IP networks (IP multicast) works the way in which a packet designed for group broadcasting is sent to a special address, which functions as a proxy for an explicitly defined group of hubs (IP addresses). The packet is delivered to all hubs which are members of a particular group. The group is formed on the dynamic basis; within this group, the hubs continuously log in and out.

Consequently, all the stations may be considered as members of one virtual LAN since the group forms one domain of omnidirectional broadcasting. Nevertheless, it significantly differs in two distinctive characteristics – the group is dynamically formed only for a limited period; thus, it is very flexible and its range is not limited by routers; i.e. it may expand, for example, to a vast WAN network.

Why did VLAN develop?
VLAN technology started to develop around 1995; at the beginning it involved only close source-based activities. Actually, these activities did not expand until a couple years ago, namely in middle-sized and large companies although an adequate standard has been defined long ago.

Main reasons for VLAN development were as follows:

- group or section-based grouping of network users or service-based instead of physical position or section-based communication between these groups
- reducing network broadcasts, which have become a big problem since a couple of years
- reducing collision domains in the time when hubs were used instead of switches
- idea of a logical grouping of users, which is mentioned in many materials, and which creates VLAN consists in
- Organizational structure – provided most of the communication is carried out within the section with printers, files servers etc. and there is no communication between individual sections; only a few services (mail) are common to all.
- Services - VLAN associates workers who use the same services (accounting, DB etc.).

7.2. Major advantages of VLAN

The way of integrating the communication in VLAN

VLAN integration is usually set up on a switch (only is special cases, a tagged communication goes through trunk from another device). Switches supporting VLAN always include at least one VLAN. It refers to a default VLAN NO 1, which is not possible to be deleted or switched off. Unless set up in a different way, all ports (i.e. the entire communication) are integrated in VLAN 1.

There are four principal methods of integrating a communication in VLAN; however, mostly the first method is employed in practice.

1. port-based method

Switch port is manually and tightly integrated (configured) in a particular VLAN. The entire combination going through this port belongs to the particular VLAN; i.e. as long as another switch is added into the port, all devices connected to that will be in the same VLAN. As a matter of fact, it is the fastest and most frequently used method. In addition, there are no specific requirements in regard to VLAN integration; its definition is local-based on each switch. Moreover, it is clearly arranged and easy to administer.

2. MAC address-based method

Ports are integrated in VLAN according to the source the MAC address. Therefore, a detailed table containing a list of MAC addresses for each device, together with a particular VLAN, needs to be drawn up. The key advantage is its highly dynamic integration; so if a device is connected to a different port, it will be automatically integrated in the appropriate VLAN.

There are two ways of using this method; either port integration in VLAN is set up according to the MAC address of the first frame – the setting remains the same until the port switches off, or each frame is separately integrated in VLAN according to the MAC address. Nevertheless, this method is very performance demanding.

However, Cisco found out a solution called VLAN Membership Policy Server (VMPS), which requires a special server administering MAC address tables. Moreover, this method integrates ports in VLAN; as a consequence, provided more devices (20 max.) are connected, all of them must be in the same VLAN.

3. Protocol-based method = according to the information from 3rd layer

This method determines protocol-based integration of the transferred packet, e.g. separation of IP operation from Apple Talk, or IP address or scope-based integration. However, it is not very common in practice. A device must contain a strictly defined IP address and the switch must operate as far as the third layer (it usually operates as far as the second), which means a considerable slow-down.

4. authentication-based method

This method verifies a user or device by IEEE 802.1x protocol, and according to the provided information, places it in VLAN. Primarily, it refers to a safety method which controls the network access (NAC); however, when expanded, it also serves to VLAN. The method is also effective due to its versatility; i.e. neither a physical device, nor a place of connection is important. RADIUS server, which verifies the user's identity, also allows mapping of VLAN users; subsequently, after a successful authentication, this information is sent. This method also allows a special setting that in case of a user not being authenticated, he is integrated in a special host VLAN.

Cisco switches may encompass a single-host port, which allows connecting only one device, or a multi-host, which, on the one hand, allows connecting more devices to one port, but after the first authentication, the port itself is authenticated as a result of which all devices may communicate.

7.3. Principles of VLAN communication

Actually, there are two situations dealing with VLAN membership. The first one deals with communication within one switch; the second one deals with communication between several switches.

One switch-based VLAN

VLAN communication within one switch is actually very easy. In its operation memory, the switch retains information about a particular communication (port) belonging to a particular VLAN; as a consequence, only one correct routing is allowed within one switch. In such a case, individual ports are integrated in one VLAN, namely, either statically, or dynamically as has been mentioned above (options 2, 3, 4). Cisco refers to these ports as access ports.

Multiple switches VLAN

As a matter of fact, a more complicated situation arises on condition that the information about a particular VLAN integration must not get lost upon transferring to another switch; i.e. in order for the entire network to use identical VLAN irrespective of a particular switch-device connection. Moreover, may be interconnected on two switches, then integrated in the same VLAN, and required information may be transferred. However, this procedure is very ineffective.

8. URL, X/HTML, HTTP

URL is an acronym of Uniform Resource Locator. It is used for correctly identifying documents on the internet. An example of URL website is

http://www.adaptic.cz/znalosti/slovnicek/url/

Where the address of our server consisting of the top-level domain (cz), second level domain (adaptic) and the third level domain (www), which are separated by dots, may be seen. Furthermore, URL contains a path to a website of a directory of the structure (/znalosti/slovníček/url/) separated by slashes. The last part of the URL is a protocol which allows asking a server for this website; in this case, protocol HTTP (ono http:// at the beginning) is to be dealt with.

In addition, URL may contain:

- a title of the website including its ending (e.g. index.htm),
- port number, which identifies a required service (written behind TLD and separated by colon, e.g.:80),
- name and password when required (written right behind the protocol – username:password:@).
- URL tabs referring to a particular place on the website (written as #tab at the very end of URL)
- website parameters (written behind the website name and separated by a question mark, e.g.? logged=true) may also be an integral part of URL.

8.1. Types of URL

The form of URL mentioned in the example above refers to the **absolute URL**; in contrast to **a relative URL**, which refer to a particular position of a current document (website containing its links).In such a case, some parts might be left out; i.e. provided a referenced website is located on the same server, the name of the server (domain) may be left out within the particular URL. On the other hand, as long as the website is located in the same directory as the referenced website, there is no need of writing a path in URL either.

A well-treated URL is called **cool URL** (also in Czech). Provided only applicability is of our interest, we refer to a so-called **user friendly URL**. On the other hand, provided only search engines are considered, **SEOfriendly URL** is to be dealt with.

8.2. The importance of URL form

URL form heavily influence the visibility and applicability of the web. For example, if a domain name is short and comprehensible, visitors are more likely to remember it and,

moreover, it is easier to be dictated via phone. On the other hand, supposing a domain name contains a keyword (or different parts of URL); it is highly useful for a search engine optimization. However, URL containing a large number of parameters works in the opposite way in both cases; i.e. URL written in capital letters or a longer URL (they are also slashed in e-mails and the e-mail program renders them illegible).

URL

URL is written either as an absolute address, or a relative one. While the absolute address precisely corresponds to URL, the relative address refers to a certain abbreviated address entry, which is based on a search engine understanding the entry from the address of the current website. Caution – URL considers capital letters.

Absolute URL

Each absolute address consists of a protocol and domain name. It is mostly followed by a directory path and file name. Sometimes, the address contains a port number, tab name and query string.

Protocols

Http or https protocols most often transfer HTML websites. These are written as http:// and https:// into URL.

All the same, a big difference does not arise as far as the server communicates with the search engine. Http is transferred un-encoded via the internet; https is an encrypted protocol.

As a matter of fact, http and https should not be swapped since both of the protocols may not work on a particular server, which means as long as absolute addresses are used, it must be found out which protocol works on the particular web.

Domains

Each internet server has its domain name. It consists of three parts separated by dots.

- name of the virtual server, mostly www (third level domain)
- second-level domain name (registration required)
- top-level domain, mostly cz, sk or com

Port

Generic domain is not usually followed by a colon or port number.

Directory path

The target file is usually stored either in a directory, or directly in the root of the server. Directories should be written in URL after the generic domain. A forward slash (not a backslash) comes before the name. Multi-level directories are written one after another separated by slashes.

Files

File name is written after the directory path (if it exists). A slash is written before the file

name.

Tabs

A direct link may connect to a tab in a referenced document. A hashtag '#' and tab name is written in URL after the file name.

Query

Input data for a certain script may also be a part of URL. These are written after the question mark at the end of URL. The syntax in name=value&name2=value2.

Example

A typical example of absolute URL may be as follows:

http://www.jakpsatweb.cz/html/url.htm#priklad

Part of the address	Example	Other possible values
protocol	http://	ftp:// , mailto: etc.
3rd level domain (server)	www.	www. , anything.
2nd level domain	jakpsat-web.	seznam. , mujweb. etc.
top-level domain	cz	com, sk, gov etc.
port	nothing	:80 , :*number*
path (directories)	/html/	/, /anything/directory/
file name	url.htm	index.html etc.html
tab	#example	#tabname
query	nothing	?variable=value

Relative addressing

As a matter of fact, writing the entire absolute address is often a needless and lengthy process. Therefore, there is a way of facilitating by means of a relative address.

The idea of relative addresses is based on existing files, which are interconnected, stored in the same server. Each file which requires another URL file has an absolute URL. For that reason, only a file path, slash and file name needs to be written in the address,

all of which means a relative URL.

Relative URL = path/file_name

Directories are separated by slashes. On condition a target file is situated higher in the directory hierarchy (thus a certain “jump-up” is required), two dots must be written for the superior directory.

Example: inserting a picture with logo “Jak psát web” (How to write a web) into this website would be carried out using a relative address as follows: <img src="../images/jakpw.png">

8.3. XHTML

XHTML refers to a modern mark-up language having replaced already old HTML language. XHTML also applied through XML language as a result of which it differs in several ways; e.g. requirement for coding declaration, more strict rules concerning entries (they must be closed) and attributes (small letters in quotation marks).

Nowadays, XHTML exists in two versions. The first is XHTML 1.0, further divided into three variants, Frameset (for websites using frames), Transitional (facilitates transition to XHTML) and Strict (the most strict variant). The second version is XHTML 1.1. As a matter of fact, it does not significantly differ from XHTML 1.0 Strict variant; it is merely divided into several modules.

Although opinions about this issue vary, Strict XHTML presents the most practical way concerning www websites. In fact, XHTML 1.1 is not compatible with older browsers; on the other hand, two remaining variants XHTML 1.0 are too unrestricted, which means, like HTML, issuing high demands on a coder’s discipline.

What is XHTML

XHTML is a different, then newer, HTML standard. By and large, HTML did not develop for a long time; it was in HTML 4.01 version when the first attempt XHTML was made.

The ‘X’ at the beginning of XHTML means eXtensible; however, practically, it rather refers to its “narrowing” and “cutting”.

On the whole, XHTML support is completely identical with that of HTML in current browsers (written in 2004 and applies to 2012 as well). Although estimated that future XHTML support would be better than that of HTML, there is no sound reason for assuming, based on experience of browser development, that it would come true.

Differences between XHTML and HTML

XHTML, in contrast to HTML, must have all the tags including unpaired tags such as <meta>, <link>,
, <hr> or <img> terminated. The entry may bear more forms; either clas-

sic (valid) <img></img> or abbreviated <img/> or slightly modified <img />. Nevertheless, the first method is not recommended to use on condition XHTML document with text/html type is sent. The second way, without a gap, is not recommended to use due to older browsers, which might leave out the last attribute as long as entered.

Furthermore, XHTML, in contrast to HTML, must have all the tags and their attributes written in small letters for the reason of case sensitivity of thus declared and referenced DTD and X (HT) ML; in other words, it considers font size. Provided an own DTD is declared, capital letters may be greatly used.

All attribute values must be put in quotation marks.

In addition, a document must begin with XML declaration, although its use is not mandatory if the document is encrypted in UTF-8, or the encryption is provided by a higher protocol (e.g. http).[14]

As long as frames are needed, XHTML 1.0 Frameset, and for individual websites XHTML 1.0 Transitional, may be declared.

XHTML document should be sent via another MIME type than common HTML documents.[15]

8.4. HTTP

HTTP refers to an internet protocol originally designed for exchanging hypertext documents between a server and browser (i.e. www service). In fact, the current HTTP version is able to transfer any files and is used for many more functions (e.g. running remote applications). There is also a secure version of HTTPS for HTTP.

HTTP is based on a query → response principle; individual queries are not discernible from the point of view of the server. Therefore, HTTP is also called stateless protocol. In fact, it was a great advantage in times of internet presentations; however, when programming more complex web applications, thorny problems occur since, for example, HTTP does not allow saving a basket content in the online store. Consequently, effective methods of breaking it, 'Cookies' for example, need to be applied.

HTTP protocol
Http protocol explores a way a browser communicates with the server while downloading websites.

Effective use of HTTP
On the whole, in order to create a website, knowing http protocol is not required. However, as soon as more complex issues arise or a more advanced search engine optimization is to be dealt with, a greater familiarity with activities between the server and client is required.

Http headers provide information about e.g. redirects, cacheting, cookies, compression or referrer.

As long as server scripts or more complex web programs are written, it is good to know where and how to send a particular http response.

The abbreviation HTTP means Hyper Text Transfer Protocol (Hypertext is a text with links).

Functioning of http protocol
A client talks to the server; i.e. client wants something and the server provides it.

- A client is usually represented by an internet browser (Explorer, Mozilla, Opera); however, a search robot or another program may represent a client as well.
- Http server involves a program running in a server room on a computer (although this computer is also called 'server', it does not refer to 'http server'). The most frequent http server is a program called Apache.

Thus, http protocol is a sort of language by which two programs talk; they talk via a network, mostly the Internet.

A client usually requires a particular website – he connects to a server and asks the server for URL websites. Such a request is formulated within HTTP protocol (the connection itself is carried out via TCP protocol). Subsequently, the server submits the request and sends back a response also written in HTTP protocol. For instance, it sends http headers to a client followed by a website text in HTML. The client receives the response, reads the headers and displays the website.

An example of http communication

For instance, a reader wants to read this website. Its URL is as follows

https://www.jakpsatweb.cz/server/http-protokol.html.

1. Reader writes this URL in the browser.
2. The browser (client) evaluates the domain and finds out via DNS which IP addresses should be asked for.
3. The client establishes a connection with the server on the found-out IP address via TCP protocol; henceforth, HTTP begins.
4. Browser sends this HTTP command to the server:

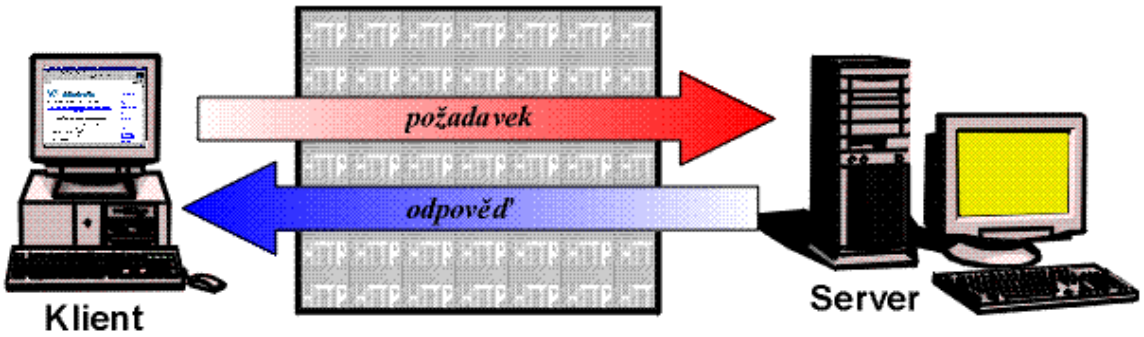
```
GET /server/http-protokol.html HTTP/1.1
HOST: www.jakpsatweb.cz
```

HTTP protocols
World-Wide Web service is based on three principal technologies – HTML, URL and HTTP. HTML refers to a mark-up language used for a standard description of the content and structure of websites. URL concerns specific addresses used on Web – each website has its own definite address in the form of URL. HTTP – Hypertext Transfer Protocol – is a protocol used for communication between browsers and web servers by means of which URL websites required by a user (via a browser) are transferred to the server. On

the other hand, the server sends back a website written in HTML to the user by HTTP protocol.

For a full understanding of CGI-scripts principles, at least a rudimentary knowledge of HTTP protocol is necessary. Therefore, basic characteristics of HTTP protocol will be looked at in the following text.

HTTP protocol results from client/server architecture. Client, in this case a browser, connects to the server and sends a request. In response, the server sends a response. The standard format of the request and response is defined within HTTP protocol. All the same, the whole situation complicates the fact that there are three protocol versions – 0.9, 1.0 and 1.1. As a result, the request and response format significantly differs in the individual versions.



Picture 1: The course of communication between a client and server

Basic characteristics of HTTP protocol

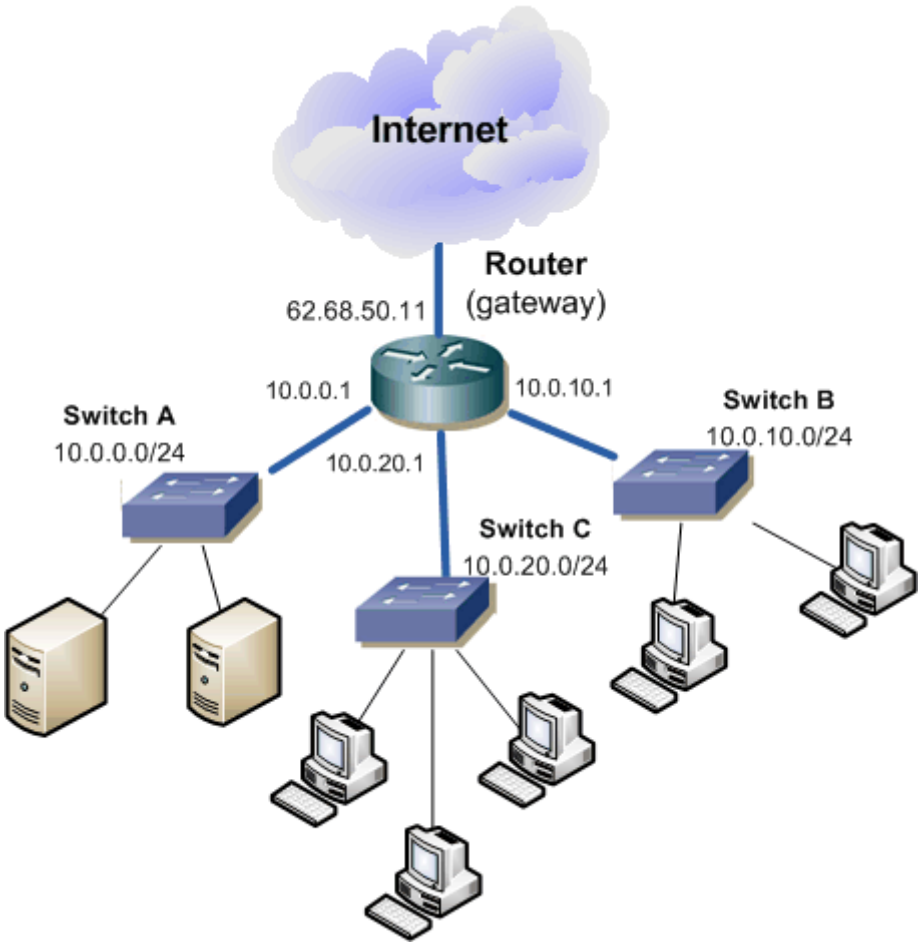
To fully understand the article, a more comprehensive knowledge of basic characteristics of HTTP is required; i.e. the actual operation of the protocol. HTTP protocol refers to an application level for distributed hyper medial information systems; i.e. this internet protocol is generally used not only for the data transfer between the client and server, but also many more operations. As a matter of fact, HTTP protocol is stateless so that it does not discriminate between clients from which it receives requests. For example, if one client sends two requests simultaneously, the server will not recognize that the same client is to be dealt with.

HTTP exists in three versions, namely 0.9, 1.0 and 1.1. The first one, referred to as HTTP/0.9, existed as a simple protocol able to transfer data on the internet in a limited way. In fact, HTTP/1.0 version allowed the protocol to transfer information in MIME format so that it could contain meta-information about the transferred data. All the same, the most significant improvement was achieved by making all connections permanent; it was implemented in HTTP/1.1 version, which is the last updated version. It means that a connection is not closed until one of the client-server pair sends a closing header. Formerly, HTTP closed a connection after each individual server response. By this huge improvement, a transfer speed significantly increased since the server does not have to open a new connection for each picture, frame and applet.

9. Routing protocols

Routing refers to a technique focused on interconnecting individual networks (subnets). The original device designed for routing was a router, but nowadays L3 switches, firewalls or merely servers/computers are applied more frequently. Router forwards communication from one network to another.

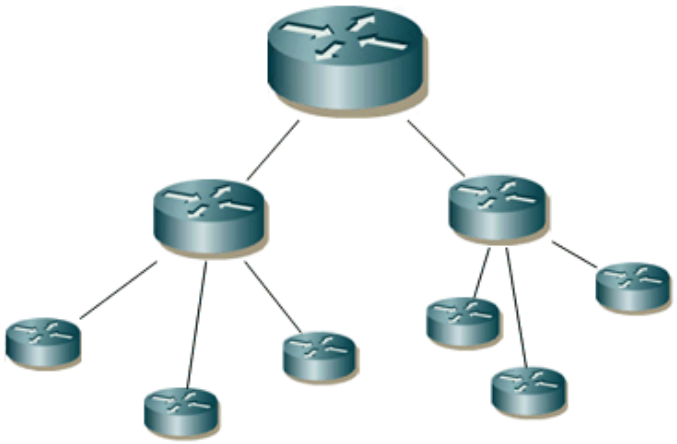
The following picture illustrates a simple example of a network with A, B and C subnets. These subnets are interconnected through a router and thus to the internet. Consequently, provided B subnet station wants to communicate with A subnet server, it sends data to the router and this secures their delivering to A subnet. On condition the station wants to communicate on the internet, the router will send the data to a different interface.



Dividing the network into subnets is based on a hierarchy and all the interconnections must have a router. The communication is carried out upwards to the nearest layer, which interconnects particular subnets, and then it goes down again. The path length is calculated by **hops,** which refers to individual transits from a device to device; i.e. a number of routers in the way + 1. A direct connection between two computers amounts

to 1 hop. A term '**next hop**' is also frequently used and refers to the address of the next router in the way.

The next picture illustrates the above-mentioned situation. There is a small (only schematic) section of a wider network or internet. Small routers are on the leaves of the tree and are connected to switches and computers. These routers associate to other (larger) ones on different levels. As a matter of fact, bigger routers are always redundant in order for the network to resist a blackout or balance the load.



9.1. Routing – technical terms

Router

A device performing routing

Routing

The process of forwarding data between networks

Route

An applied path written in a routing table

Routing table

Contains records on individual routes

Routing protocol

Manages directing of a routed protocol; defines the best route towards the target and sends routing information to other routers

Routed protocol

IP, IPX or Apple Talk

Eventually, one more important and frequently used term

Router on stick

refers to a router connected to a switch by one trunk port; i.e. only one router and one line are available, which causes a considerable strain on both, the router and line and brings about failure problems

Division of routing protocols

The routing table shows several records on routes, which depends on their origin. Accordingly, packets are routed in one of the alternative ways of routing:

- static routing – manually entered routes (records in the routing table), secure and high-quality, but it does not reflect changes in the network topology
- dynamic routing – the network is automatically adapted to changes in topology and transport; routes are automatically calculated by a routing protocol
- default routing – unless there is another way, default routing is applied

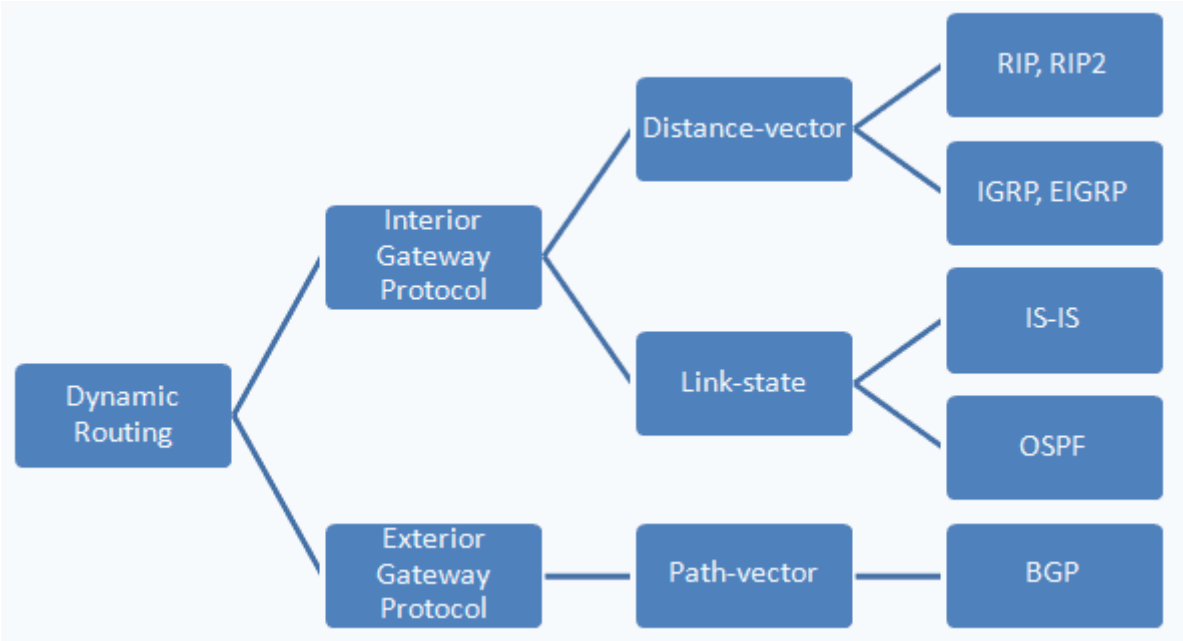
Dynamic routing protocols are divided into two distinct types

- distance-vector routing protocol – routers maintain the routing table containing information about the distance (vector) from a particular network; furthermore, they send the routing table to their neighbours, which draw up their own routing table which is further sent on; for calculating the most convenient route, one (number of hops at RIP) or more metrics (line throughput and IGRP delay) are used. In addition, an upgraded type of a distance-vector protocol is represented by a path-vector protocol.
- Link-state routing protocol – routers maintain a comprehensive network topology database (created by means of LSA), change link-state advertisements (LSA); LSA are caused by a certain event on the network. This protocol also sends Hello packets, which contain information about the protocol, quickly responds to changes of topology; all the same, it expands through a greater range and uses more resources on the router; moreover, its metric is complex-based and the most convenient route is calculated by Dijkster algorithm Shortest Path First (SPF).

Note: actually, there is one distinct type based on distance-vector protocol which has some characteristics of link-state protocol; it concerns a hybrid routing protocol or advanced distance-vector protocol. Its only representative is EIGRP.

Furthermore, dynamic protocols are divided thus; whether they are designed for being set inside a local network (to put it more precisely, inside an autonomous system (AS), which may consist of several LAN), or they operate throughout networks (they link AS together).

- interior gateway protocol - IGP - routing inside Autonomous System (AS)
- exterior gateway protocol - EGP – routing throughout AS


Dynamic Routing
Interior Gateway Protocol
Exterior Gateway Protocol
Distance-vector
Link-state
Path-vector
RIP, RIP2
IGRP, EIGRP
IS-IS
OSPF
BGP

10. Security and encryption

10.1. Network security

Risks:

eavesdropping, modification of transmitted data, unauthorized access to a local network

Adequate protection:

- Protecting data from unauthorized acquiring, exchanging or deleting
- Computing capacity of individual hubs
- Cutting back on functioning or disturbing the service traffic

Passive attacks

- "Eavesdropping" in order to get unpublished information which might be abused
- Traffic monitoring – analyses of thus operated contacts

Active attacks

- Data modification
- Providing false data
- Active attacks cannot be 100% avoided; however, in contrast to passive attacks, they are easier to detect

Objectives of security services

- Securing data confidentiality – by means of encryption of the entire communication channel, or selected sensitive data
- Securing authentication of network users (revealing a masked intruder)
- Securing data integrity
- Securing the refusal of messages – to secure the impossibility for a user to deny sending the message and for the receiver to deny receiving the message
- Allocation of access rights in order to control the access to the computer, data and applications; identification and authentication of applicants for access are also its integral parts
- Securing the availability of network services; attacks on the availability of services may be prevented by authentication and encryption

10.2. Firewall

- It refers to a set of measures (implemented by HW and SW) which protect the network against an unauthorised access from the outside and, simultaneously, against information leakage
- It allows, e.g. controlling the user access from external and internal networks, setting up access rights, filtering out dangerous services, channelling security to one communication

hub, blocking a hostile mapping of the internal network, auditing legal and illegal operations.

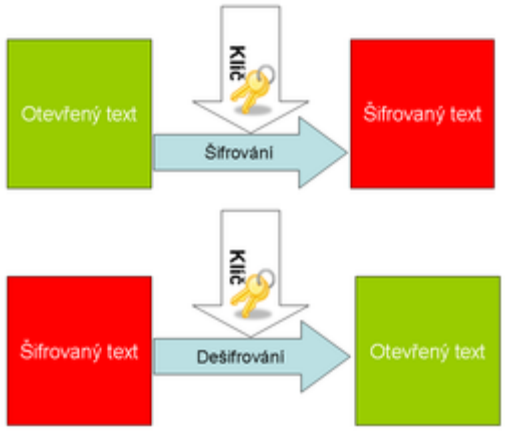
- It provides security when entering and exiting the network
- It serves as a filter; i.e. it decides what and to where it will be allowed.

10.3. Encryption

1. Symmetric encryption

Encryption and deciphering using a single key. Symmetric encryption, sometimes also called 'conventional', refers to a cryptographic algorithm that uses a single key for encryption and deciphering. Thus, it differs from public key-based algorithms using a pair of keys – a secret and public one.

The great advantage of symmetric ciphers consists in their low calculation complexity. Public key-based algorithms may be hundreds of times slower. On the other hand, the main disadvantage consists in the necessity of sharing a secret key; therefore, the sender and receiver of the secret message must agree on a particular secret key.



Picture 2: symmetric encryption

Open text → *encryption* → *encrypted text*

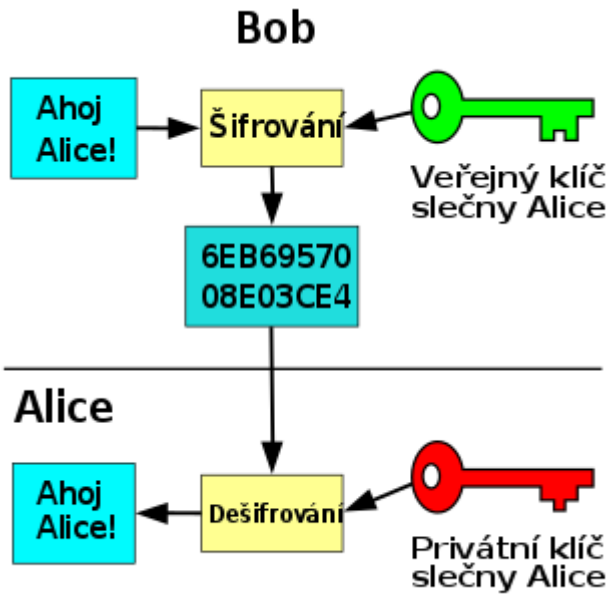
Encryption → *deciphering* → *open text*

'klíč' – 'key'

2. Asymmetric encryption

Examples of asymmetric encryption. **Asymmetric encryption (public key-based encryption)** concerns a group of cryptographic methods which use **different** keys for encryption and deciphering. On the whole, this is the main difference from the symmetric encryption, which uses a single key for encryption and deciphering.

In addition, apart from an obvious possibility of a secret communication, asymmetric encryption is also used for an electronic signature; i.e. a possibility of revealing the author of the data.



Picture 3: asymmetric encryption

Bob

Hello, Alice → encryption ← Alice's public key

Alice

Hello, Alice ← deciphering ← Alice's private key

Encryption key for asymmetric encryption consists of two parts; the first part is used for message encryption (the receiver of the message does not have to know this part); the second is used for deciphering (the sender of the encrypted message usually does not know it). Obviously, the one who encrypts does not have to share any secrets with the receiver who deciphers as a result of which there is no need of exchanging keys; this is the main advantage of asymmetric encryption.

The most common version of asymmetric encryption consists in using a public and private key; encryption key is public; the owner of the key publishes it so that whoever may encrypt the specified messages; on the other hand, the deciphering key is private; i.e. the owner keeps it secret and uses it for deciphering the messages (there are also other methods of asymmetric encryption which require to keep the key secret).

Obviously, the encryption key 'e' and deciphering key 'd' must be mathematically intertwined; however, in regard to the encryption effectiveness, there is impossibility of calculating the deciphering key from the encryption key.

Computer network security

In this case, the term 'network' refers to a system of several computing systems; users access the network via one of these systems.

- Sharing - a very large number of people may potentially access the network; various devices may be controlled by not necessarily secured systems

- Complexity - the network contains various operating systems communicating via a connecting mechanism, which should provide the security; however, this mechanism must be rather universal; moreover, the network as a whole cannot be subjected to testing or, even, certification.
- Unknown perimeter - we do not know all connected people; moreover, it is not clear how other devices work
- Number of vulnerable places - as a matter of fact, it is necessary to trust the security mechanisms in all the devices since many network parts are located off the operators' supervision
- Unknown path - in most cases, it is not possible to control the path of data transfer; thus, anyone may capture them without previous information. Anyway, communication protection may be provided as follows:
    - Data flow - also referred to as 'stream enciphering'; i.e. encryption of data in the way that conveys the impression of a communication channel being highly reliable with respect to a possible attack
    - Individual messages - correspond to a today's modern free binding using 'messaging'; application messages are encrypted, or encryption of the data flow is carried out either between two network hubs, or between two applications running on these hubs. Concerning Link Encryption, data are encrypted just before the entry to the communication medium, and then deciphered right after arriving at the second computer. This encryption is carried out on the level of physical layer or data-link layer of the reference model. The key advantage is that this mechanism is very transparent to the user; moreover, it may be very fast and easy to connect to other devices.

End-to-End Encryption

It provides a cryptographic protection throughout the transfer process. This encryption is carried out on the level of application or presentation layer of the reference model. All the same, the encryption is no longer transparent and, in order to be effective, it must be appropriately integrated in the entire system. Its main advantage is that there is no need of encrypting the entire communication but sensitive data. In contrast to the link encryption, it provides authentication and integrity (end-to-end). Sometimes, both methods are used simultaneously - link encryption for a common preventive data protection and end-to-end encryption for achieving a high-quality protection of sensitive data. In relation to the encryption, there is a necessity of distribution mechanism and administration of necessary encryption keys, competent authorities for securing the system operation of cryptographic protection and suitable cryptographic devices providing basic functions of the cryptographic protection. Apart from common problems of access control, other thorny issues arise.

Graded access rights

The access to sensitive data may be confined only to some network hubs. On condition an authorized user asks for an access from a different hub, his access rights may be severely limited, or the access to data may be entirely denied. After receiving the call, the

silent modem does not immediately start generating, but waits until the other side tries to ‘negotiate’. Thus, the access is, to some extent, limited only to users who know that the line leading the computer is to be dealt with. Furthermore, the method limits the possibility of a random location of this port. In case of IP protocol, a service available on a particular device on a different port from the usual one may be an alternative for a silent modem.

11. Peer-2-peer networks

During last 10 years, exchange networks of peer-to-peer type increased in importance in the area of downloading files from the internet; nowadays, the most popular one is Bittorent protocol network, which has tens of millions users and tens of clients (µTorrent, Vuze/Azureus). Furthermore, in last 5 years, a recent phenomenon is represented by an increasing popularity of public commercial web servers for sharing files (filehosting), out of which Rapidshare.com is the most popular one; however, systems encouraging cooperation between more people on one document, or synchronizing files (Google Docs resp. Humyo.cz) may be included as well.

Peer-to-peer networks

The term peer-to-peer network (P2P) is rather general. As a matter of fact, it involves any network where there is a symmetric communication or interaction between computers (all of them can initiate, or on the contrary, on the grounds of external initiation/requirement, carry out necessary transactions and operations). Therefore, it is not possible to functionally divide its hubs into two common types - clients and servers. As a result, a basic central task of the server is failed; subsequently, one hybrid universal type of computer hub, so called 'servent', arises (the term 'client' is also frequently used). Decentralization of the internet, which began in the age of Usenet (1979) and Fidonet (1984, BBS systems), has been successfully accomplished by P2P networks. However, the equality of all computers in the network does not mean that the network is homogeneous in all the hubs in regard to quantitative parameters such as connection speed, mass of shared data, local configuration, processing speed etc.

On the whole, P2P label has almost become a medial synonym of internet exchange networks (i.e. systems of sharing files) recently although a lot of different types of applications may work on this principle - for example, applications for distributed calculations, spreading news, voice communication and chat (IRC, Skype, Qnext, DKMessenger) or P2P internet radio broadcasting and TV stations. As a matter of fact, medial stereotypes identified P2P networks (sharing files) with piracy. Without trying to find out the truth, it is necessary to observe that e.g. Bittorent network is efficiently used for a legal distribution of large programs and files (Linux distribution); furthermore, there are torrent catalogues and trackers which help distributing and registering only legal (e.g. film) material of a public domain quality (http://www.legittorrents.info/, http://beta.legaltorrents.com/, http://www.publicdomaintorrents.com/, http://www.jamendo.com/en/); i.e. the one which is legally free of charge.

A distinguishing mark of P2P exchange networks is that files or text messages saved on any computer connected to the internet (equipped with a running client/server of a particular network) may be shared within these networks. On the other hand, networks technically differ to the extent of the real degree of functional symmetry, i.e. homogeneity and decentralization, and relative "anonymity". In fact, a perfect anonymity is practically impossible on the internet. Different generations of these networks are distinguished by these distinguishing marks.

11.1. Generations of P2P networks

The first generations is represented by networks which saved file and computer lists and file and computer addresses on one, or more special central servers (Napster, OpenNap, chat - IRC (since 1988)/IRC@find, Soulseek - the latest works to this day and is frequently used by a smaller community of music fans). In regard to the evolution, the oldest exchange P2P networks use a centralized structure of client/server type for some of their activities. It mainly deals with a network connection and searching for resources (files). Thus, servers are not needed for a routine communication and sharing between end-to-end clients; such operations are carried out by peer-to-peer.

Networks of the second generation are most frequently used nowadays. Central servers are completely missing; all the same, these networks are usually not based on entirely equal servent-hubs; i.e. computers cannot be largely replaced by one another (except for such as Gnutella or Freenet). As a matter of fact, a perfect symmetry is often reduced in practice within peer-to-peer principles in order to increase effectiveness and reliability of searching or facilitate identifying files and thus speed up their downloading.

Today's P2P systems are characterized by 'local-central' or specially-delegated/dedicated components such as various 'super-hubs' (FastTrack network with clients such as Kazaa), hubs (DirectConnect, DC++) or search engines and indexators (OpenFT), of which may (voluntarily) become all individual computers. These special 'super-hubs' are important for their quick connection and cataloguing all other ordinary computer-hubs and their resources and preventing occurrence of 'tight throats' within data flows. The next 'asymmetric' components refer to specialized storage servers providing digital hash imprints of individual files (identification signature, e.g. torrents-BitTorrent, line magnets - mainly Gnutella, ed2k - eDonkey/Overnet lines); i.e. servers which help coordinating a sped-up 'swim' downloading of individual partial data segments between computers.

Despite this fact, newer networks use peer-to-peer structure for almost all purposes and tasks; therefore, they are also called 'real P2P networks'. Moreover, real P2P networks have a distinctive characteristic that their total transmission or communication capacity of a common user increases with the number of users/hubs; in contrast to the decreasing character of centralized systems, the fact of which is supported by technologies of segment downloading.

In addition, multi-protocol clients, which are able to search and download files within more networks (protocols) - sometimes simultaneously, are also an essential feature of the second generation; for example, these are MLDonkey,KCeasy and Shareaza. Furthermore, the integral part are so called 'overlay protocols' forming a certain 'supernetwork' within the particular network (Overnet).

Emerging of the third generation of networks and clients introduces or enhances encryption elements (masking of data traffic, which had been included in BitTorrent network), anonymity or pseudonymity; i.e. secreting IP addresses of computers/hubs (encryption of mutual routing or linking). Moreover, it seeks for a larger decentralization than nowadays (Freenet, GNUnet). Actually, some of these networks cannot be accessed

without approving of other users (networks such as friend-to-friend - ANts P2P, WASTE, MUTE). Others almost have features of so called virtual private networks (VPN); e.g. I2P, which allows running of all internet activities in private. Sometimes, these networks include not only a mere file sharing, but also emphasize a protected area of communication and publication of documents without a censorship interference as a result of which they become decentralized and protected by a private groupware. On the other hand, the main disadvantage of these systems consists in a smaller user's comfort and relative slowness.

The above-discussed internet radios and TV broadcastings (internet streams) of peer-to-peer type sometimes belong to P2P applications of the fourth generation (TVUPlayer, PPLive, PeerCast, PPStream). A certain part of P2P-TV systems is based on BitTorent protocol or similar ones (in this case, however, television 'on-demand' is to be dealt with; not a real-time streaming); all the same, the main ones use their own, different, systems. On the whole, from tens up to hundreds of TV channels (often sport) are broadcasted in this way with a high efficiency (given to the fact that each receiving hub may simultaneously be a retranslation transmitter for other hubs.

11.2. Filehosting

On the other hand, filehosting (both - public and commercial) on web servers refers to a rather conservative and technologically old service yet very effective, considering today's connection speed. In this area, there are over 100 important foreign servers and, probably, up to 10 Czech ones. Conditions and systems of their use individually varies in regard to downloading and uploading; however, they often have two different downloading modes; i.e. capacity or time limited free downloading (paid by an advertisement and limited via OCR/captcha systems, temporary lines and time intervals) and "bonus" downloading, which is less limited, or practically unlimited only for a small user fee. A limited selection of these servers, without domain suffixes, which, in case of interest, may be added by a reader himself, is suggested below.

Foreign:

Rapidshare, depositfiles, filefactory, megaupload, mediafire, sendspace, uploading, zshare, ifolder, hotfile, icefile, letitbit, filefront, ifile, easy-share.

Czech:

Edisk, uloz.to, czshare, leteckaposta, quickshare, bagruj, nahraj.

In this area, there are a few programs which may, to some extent, automate and thus facilitate downloading from these servers, e.g. Universal Share Downloader (USD) or RapGet.

Other important links:

- www.filesharing.eu Filesharing via P2P networks

- www.slyck.com News and Slyck portal
- www.zeropaid.com News and Zeropaid portal
- www.filesharingz.com News
- www.p2pnet.net News
- www.p2pforums.com Discussion and news portal
- www.infoanarchy.org "Decentralizing" P2P wiki-portal
- www.openp2p.com O´Reilly's website, focusing on P2P area
- www.planetpeer.de Portal for anonymous networks not only for filesharing (MUTE, I2P, Freenet etc.)
- www.fileshareworld.com Signpost for exchanging systems and P2P clients
- www.ftc.gov/bcp/workshops/filesharing Opinions of the US Federal Trade Commission on filesharing
- http://cs.wikipedia.org/wiki/Peer-to-peer Czech password for Wikipedia
- http://p2ptv.yourglobaltv.com/ a http://www.tvavailable.com/P2P_TV/ Overviews of the most frequent P2P TV clients
- http://en.wikipedia.org/wiki/P2PTV Seminal article on P2P TV
- http://www.yourglobaltv.com/p2pchannels/ Main sport channels of P2P TV
- http://en.wikipedia.org/wiki/File_hosting_service Hosting servers in general
- http://www.dimonius.ru/dusd.php - Universal Share Downloader (USD) - automatic downloader
- http://www.rapget.com/en/ RapGet – similar to the above-mentioned USD.

12. Anonymity on the internet

Anonymity on the internet has recently been a thoroughly discussed topic even by governments of developed countries. Anonymity has always been an important factor both in past and present. Nevertheless, along with the arrival of new technologies and their subsequent penetration in the society, this key topic should be more extensively discussed than ever before. On the one hand, individual countries have taken bold steps to assume control over the internet in order to identify the user due to a greater security (main concerns include terrorism, child pornography, and drug sale or money laundry). All the same, on the other hand, countries try to protect the privacy and sensitive personal data of internet users, which has been supported by enforcement of so called 'Cookie Law' by the EU.

In order to achieve a perfect social anonymity, it is necessary to ignore all seven dimensions of identification information:

- Personal name
- Location
- Pseudonym connected to a real name or location
- Pseudonym revealing other information
- Revealing behavioural patterns
- Membership in a particular social group
- Information, object or skills showing personal characteristics

Reasons for anonymity remain the same in both, the real environment as well as in the virtual one - avoiding consequences of one's own behaviour (fear of repressions, misunderstanding etc.)

Identification technologies

Modern technologies offer a lot of ways and methods of identifying the individual.

IP address

It more or less refers to a unique address of each device connected to the network. For example, IP address may reveal information about a geographical location.

Geolocation

Recently, the use of techniques which are capable of determining the location up to one meter has significantly increased, e.g.:

- **Constraint-based geolocation** is based on actively measuring the distance by a response.
- **GPS** mostly runs on mobile phones.
- **A content analysis of web posts (social networks)** refers to a method of locating users based on an analysis of publicly available posts e.g. according to cities, countries or weather in the post content. Moreover, user's activity in a view of time for locating his time zone is taken into consideration.

Cookies

Cookies serve as a permanent identifier between a client and server. However, due to their intrusion of privacy, this issue is severely dealt with by the European Union.

PRISM concerns a government project of the US secret agency NSA. This project has unlimited access to data of Google, Microsoft, Yahoo and many others with the view to the fact that the above-mentioned corporations constitute 98% of the data. Nevertheless, these corporations deny having the unlimited access to these data.

12.1. Privacy supporting technologies

Tor refers to a project based on Onion Routing concept, and thus secures user's anonymity while browsing on the internet. The program is designed for protection of personal data of users, their freedom, privacy and opportunity for a private trade in the way that protects them from tracking their activities on the internet. Nevertheless, this software may be also used for illegal activities as the use of this technology greatly complicates hunting down the offender.

Orbot refers to a Tor version for mobile phones with Android operating system.

Freenet Project - this technology is considered as a perfectly safe, anonymous and decentralized platform for sharing data directed against censorship. Each user may allow a certain space on his local disc, which is subsequently available to the entire network, for saving encrypted files from other users. In this way, an own private server is not required.

The internet comprises an extensive and complex network of communication channels, via which our data flow. Each **data packet** (hereinafter 'packet'), apart from the content itself meant to be sent somewhere, contains particular distinguishing marks, which are of a remarkable interest for "big brothers". These include **Google, Microsoft, Facebook, Twitter** etc.

As a matter of fact, they use **distinguishing marks** of our data in order to find out which websites are of our interest, the frequency of our visits, and also, our location. Subsequently, this valuable information is evaluated for targeted advertising. In addition, with the arrival of social networks, spying has gone much further.

Big brothers may successfully **combine** our web activities with those of our friends and thus draw a detailed map of the internet users. This **voluntary loss of privacy** is a price of providing services for "free". On the other hand, so as not to show only drawbacks of these service providers, it is necessary to say that their services display high qualities and elaborateness although they may not be evident at the first sight.

12.2. Basic principles of preserving anonymity

As has been said, the internet tracks mainly data about where our data come from and whether they belong to us. Thus, the major effort should be **masking these data**. As a matter of fact, VPN (Virtual Private Networks), Proxy servers or special internet browsers may serve the purpose. The advantages and disadvantages of individual solutions are suggested below.

Connection via VPN

Virtual private networks are mostly used at work, where they enable more computers to access network storage, printers or other servers. At the same time, they provide communication with the public internet. Furthermore, **our packets are marked by IP address** of VPN server, not our computer, in order for websites and big brothers to see them.

As long as we log in a VPN network which is located in a different country, our physical location will not be found out. Moreover, VPN server encrypts the communication so that nobody, except for end-to-end server, will be able to read our data.

There are many VPN from different places of the world which are meant for an anonymous internet surfing. Unfortunately, all of them are obliged to publish our IP address on condition they are ordered to do so by court; otherwise, they would be banned. Therefore, we cannot rely on the fact that if we subscribe VPN, nobody will be able to track down our IP address.

Connection via proxy server

Like VPN, also a proxy server **provides communication between our PC and the internet**; therefore, the internet will consider our data as belonging to the proxy server. However, in contrast to VPN, data **will not be encrypted**. Since proxy servers suggest a simpler solution than VPN, it is also possible to use proxy servers with publicly accessible web interface.

As a matter of fact, they are easily applicable when **a short-term application** is to be dealt with; the only thing is to put the appropriate address in their address bar. In this way, the proxy server provides the communication with the particular web.

All the same, it must be taken into account that all our data pass through the proxy server; i.e. in case of unreliable proxy, **the data may be abused**. Therefore, web proxy servers are not recommended to be used for visiting secured webs (http.).

Special web browsers and search engines

Anonymous web browsers focus on **layman's expectations from his common browser** when the anonymous mode is switched on; this mode does not hinder our internet activities from big brothers by any means - only from other users of the same computer. Thus, the entire communication is, as usual, tracked down.

On the contrary, anonymous web browsers are programmed in the way of preventing big brothers from tracking our data. Although this solution is not remarkably effective, it is a simple and quick resolution. In addition, using a browser with an anonymous internet search engine is highly convenient, instead of Google, Yahoo! or Bing, since this browser does not collect information on things which a user had been looking for.

The Onion Routing – Tor

Tor is a web browser and also a server network. It is often referred to as **the internet within the internet**. It works on a principle of the **onion routing** in which each packet is encrypted in layers - like an onion - and each server is able to decipher only one layer.

In this way, each packet passes through a certain number (at least three) of servers before it reaches its destination. Nevertheless, each Tor server knows only addresses of the incoming and outgoing server - not the entire cascade.

Therefore, **it is almost impossible to track down a packet as far as its home IP address**. Tor also gained an unsavoury reputation for allowing access to the black internet, which is associated with illegal activities. The key disadvantage of using Tor browser is its low speed since data travel, literally, to and fro all over the world.

12.3. TOR - totally anonymous internet

Digital footprint

Each global network user leaves a footprint of his activity; such a phenomenon has been called a 'digital footprint' since 1980s. To some extent, the footprint intensity, at least in the first phase, depends on us - our browser settings or proxy server installation, which provides communication between the internal network and internet and thus enables a large number of security settings. On the whole, it does not guarantee a total anonymity, but gives an adequate privacy protection for a common use of the internet.

A more effective protection

At the beginning of this century, a meticulous attention was drawn to the internet anonymity. A few projects in order to provide users with an adequate privacy protection were set up. As a matter of fact, a project of Tor network (The Onion Routing) was paid a special attention since it was funded and sponsored by American government agency Naval Research Laboratory. In the beginning, it worked as a protection of the government communication in the US. Likewise, Tor received a financial support from a non-profit organization Electronic Frontier Foundation, which deals with protection of rights in the digital world.

Hub next to the hub

How does an entirely anonymous network work? The security principle is based on a large number of hubs through which the data passes. The entire communication is asymmetrically encrypted and servers share public keys and exchange generated dynamic keys for transmitting data strings with one another. Each hub is entirely autonomous and gets the information about where to send the string on only from the message header. In this way, its knowledge of the data flow ends at the next hub. To put it more simply, the server knows to whom a particular thing should be sent and to whom the response should be returned; however, other activities are beyond its capability. The more hubs are in operation, the higher anonymity is guaranteed. The network user him-

self chooses the appropriate path and hubs through which the data should be sent.

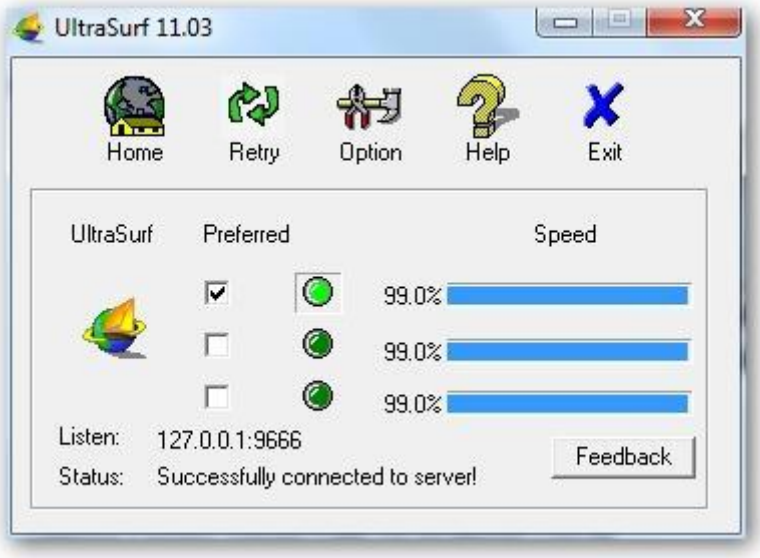
False footprints

The complexity of data transfer via anonymous servers would, at the first sight, seem highly satisfactory; however, Tor goes further. Namely, servers may send packets in a different order than have been received or false packets, which may divert a possible tracking of the data flow from the actual path, may be put in between transmitted data strings. The main disadvantage is slow response from Tor network.

In addition, there are situation in which a user, in order to protect his privacy or access some services, has to change his IP address. Nevertheless, it is not a delicate operation; a lot of services offer a solution via VPN and their own client with extra functions, an access to their own servers as a result of which the communication is redirected in the way that a user browses on the internet either anonymously, or encrypted.

UltraSurf

UltraSurf is a minimalistic program allowing anonymous browsing on the internet even to absolute beginners. This is mostly achieved by simple settings, control, low-cost operations and free availability. The program in the small window offers a connection to three different servers.



The successful change of the IP address is announced by an icon in a shape of a padlock in the toolbar; furthermore, it is possible to switch on and off controls by means of shortcuts, start the browser, automatically clear cookies and history and also set up the proxy server. In order to access American servers and services which require the local IP address, UltraSurf uses 'freekarol' reader as a result of which it belongs to the best of free-of-charge applications.

proXPN

This VPN client is very popular and widely used since it offers a fine performance, high-quality services, choice between American, British, Dutch and Singapour IP address, en-

cryption, unlimited data transfer, limited information storage about the connection (two weeks) and a lot of other things. The program may be applied with above-mentioned services for free, or an extra charge for bonus features.

TunnelBear VPN

Users often refer to TunnelBear service and VPN application as a beautiful, very simple and user-friendly application. Moreover, it is possible to use a free-of-charge version in which 500MB may be pulled through the "tunnel" for free each month; in addition, when advertised on Twitter, 1 GB extra, together with a paid and unlimited data version for five dollars a month may, be obtained.

CyberGhost VPN

The next recommended VPN is CyberGhost, which requires an encrypted connection (1024 bit SSL 128 bit AES password) to a virtual private network VPN by a software client through which further communication passes to the internet. Program is easy to control and only a few clicks enable the connection.

CyberGhost server network is reasonably frequent and efficient. Moreover, the loss of speed, in contrast to other solutions, is significantly minor. The program and service may be also used for free although with functional limits (1 GB of transferred data per month, the speed up to 2 MB/s etc.). In addition, authors offer different paid and efficiency-graded versions. CyberGhost was recommended by *Suslikus*.

Hide My Ass

Hide My Ass is a world-known VPN which has a wide user base and wide selection of services and servers. Hide My Ass offers almost 40,000 IP addresses to hide behind in 53 different countries. As a matter of fact, free-of-charge solutions cannot compete with such a large number of IP addresses so that over 11 dollars need to be paid for a month's use; however, the price may be reduced to six and half dollars if a long-term subscription is to be dealt with.

13. Attack and defense on the internet

13.1. Attack and defence of the PC

13.1.1. ATTACK

Virus

The name is derived from the analogy with biological origins. A virus is capable of self-regulation, i.e. reproduction of itself on condition there is an executable host to which it is connected. A host may involve executable files, system disc areas or files which cannot be executed directly, but by means of specific applications (Microsoft Word documents, Visual Basic Scripts etc.). As soon as the host has been executed, a virus code is simultaneously deciphered. In this moment, the virus usually tries to ensure a further self-reproduction, namely by connecting to other suitable executable hosts.

Trojan horses

In contrast to viruses, this type of harmful code is not capable of self-reproduction and file infection. Trojan horse mostly appears to be an executable EXE type of file which contains nothing (useful) but a mere body of trojan horse. Henceforth, considering that trojan horse is not connected to any host, there is only one possible form of disinfection - deleting the infected file. The older definitions say that although trojan horse appears to be useful, it is actually particularly harmful. Long ago, a trojan horse looking like McAfee virusScan program appeared although it actually disposed of files on the hard disc. Currently, several forms may be met:

- Password-stealing Trojan (PWS)
- Destructive Trojan
- Backdoor
- Proxy Trojan

Worm

Originally, the term 'worm' referred to 'Morris worm', which, in 1989, infested a considerable part of the then network, which later developed to the internet. This one and other worms (the most recent ones are Code Red, SQL Slammer, Lovsan / Blaster, Sasser etc.) work on a lower network level than common viruses. They do not spread in a form of infected files, but network packets. The packets are routed from a successfully infected system to other systems of the internet (either randomly, or by a particular key). Provided such a packet reaches a system with a specific security hole, it may be infected, and thus other 'worm packets' may be produced. As a matter of fact, spreading a worm is based on abusing particular security holes of the operating system; its success depends on the frequency of the software containing abusive security hole. As may be derived from the above characteristics, worms cannot be detected by a common form of

the antivirus software. In addition, its side effect may be a serious network infection, including enterprise LAN. The term 'worm' is often associated with a kind of infiltration spreading through electronic mail. In this way, terms 'virus' and 'worm' may overlap.

Spyware

Spyware refers to a program which uses the internet for sending data from a computer without user's knowledge. In contrast to the backdoor, only statistic data such as an overview of visited websites or installed programs are stolen. This activity is defended by trying to find out user's needs or interests and use the information for targeted advertising. However, no one can guarantee that the information or technology cannot be abused. Therefore, there are a lot of users indignant by a mere existence and legality of Spyware. Of importance also might be that Spyware spreads together with a large number of shareware programs and with a full approval of their authors.

Adware

It concerns a product obstructing a work with PC advertising. The typical symptoms are pop-up advertising windows while surfing, together with imposing websites (e.g. default website of Internet Explorer) which are of no interest to a user. A part of Adware is accompanied by 'EULA' - End User License Agreement. In this way, in many cases, the user has to agree with the installation. Adware may be a part of particular products (e.g. BSPlayer). Although the advertising provides a company during the entire activity with a particular program, its reward is a larger number of useful functions which a common free-of-charge version (advertising-free) does not include.

Dialler

Dialler involves a program which changes the way of internet access via modems. Instead of a common phone number for the internet connection, it redirects the dialling to numbers with a specific tarification, e.g. 60 Kč/minute. All the same, the era of diallers concerns only analogue phone lines (dial-up) and does not involve ADSL or other modern technologies.

Phishing

This term refers to fraudulent e-mails in which fraudulent mails looking like information from a major institution (mostly banks) are sent to a large number of addresses. These mails extensively employ 'social engineering'; i.e. the receiver is informed about a supposed necessity of filing information into an elaborated form otherwise his bank account will be blocked (in case of a bank) or alternatively, he can be disadvantaged in another way. The e-mail usually contains a link to websites with the elaborated form which as though referred to a server of this important institution. Actually, the user is redirected to an alien server created in the same design like websites of the original and true institution. Thus caught user may not spot the difference and may fill in default boxes in which confidential information, account numbers, passwords for internet banking, pin for the payment etc. are required. Thus obtained information may be easily abused by fraudsters.

13.1.2. PROTECTION

Unfortunately, the term 'prevention' is not much considered by PC users. Nevertheless, it is necessary to realize that merely installed antivirus software (even a regularly updated and correctly set) is an insufficient prevention. **Regular updates** of at least those products associated with a computer network of the internet are absolutely vital. To put it in the nutshell, it requires:

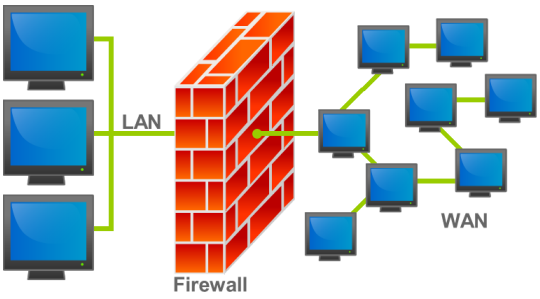
- allowing regular updates of Windows operating system and its integral parts (Start / Control panels / Automated updates)
- Regular updating of other commonly used products which might be abused from the internet (Mozilla, Firefox, ICQ, DC++); alternatively, allowing regular updates. Obviously, it also concerns various 'plugins' of particular browsers (java, shockwave, flash player)

Antivirus software

Antivirus software should be an integral part of each PC. In most cases, an essential part of antivirus software is also anti spyware. Antivirus and anti-spyware are necessary to be regularly automatically updated. In regard to the high speed of spreading bugs (mainly via electronic mail), every half an hour's update is not a luxury. On the whole, the more often the antivirus software updates the better. Likewise, it is also necessary to ensure module runs securing a continuous inspection of files manipulated by a user or application (opening /saving / running etc.). Such modules are usually referred to as 'resident shield' or, technically, 'on-access scanner'. It is also necessary to take into account that this module is the most important part of antivirus software. Although, in most cases, a module which directly inspects downloaded or sent mail is offered, it may be declined since a possible infection would be anyway prevented by on-access scanner (except for a few exceptions caused by abuse of bugs in the program). Although the virus would not be identified before put in the inbox, it would be identified at the moment of the user's attempt of opening - running the infected email attachment.

Firewall

Firewall, mostly personal firewall, is a useful supplement for the user accessing the internet. Firewall is almost vital in the moment of a user being connected via IP address and his computer thus accessible from anywhere with the internet access (a user usually gets public IP address upon dial-up connection or connection via the cable internet - Chello/UPC. Like antivirus softwares, firewall requires an appropriate use. In this way, the user should be able to discriminate between legal and illegal communication; i.e. what should be permitted or forbidden. Otherwise, firewall lacks its purpose irrespective of using a Windows integrated firewall, or a different paid firewall.



13.2. Dangerous network - resisting the attacks

As a matter of fact, networks are full of danger. There are thousands of applications that may be theoretically and practically abused. Some dangers can be relatively easily eliminated, for example, by observing appropriate safety measures such as strong passwords or up-to-date software versions. However, there are certain dangers which cannot be easily eliminated and their identification is not easy at all. These dangers will be discussed in the following text.

It mainly concerns attacks using TCP/IP protocol. This protocol is over 30 years old and it was found out that not everything had been designed and laid out correctly. On the other hand, conditions in which TCP/IP protocol was designed and created also need to be considered. As a matter of fact, its development did not foresee that within several decades the global computer network would have hundreds of millions users. However, with an utmost caution and observing appropriate measures, also the following dangers may be eliminated.

Network traffic monitoring

Network traffic monitoring is usually referred to as 'sniffing'. Basically, a network traffic analysis is to be dealt with. In the following text, basic principles and conditions on the grounds of which a possible attack may happen will be formulated. Above all, it needs to be pointed out that this type of attacks usually occurs in LAN networks. The next condition is laid down by network not using a switched connection (this condition can be disregarded - see the further text). Now, let's see how the communication in LAN network is carried out. All the data, irrespective of from or to whom they were sent, pass through a transfer medium (cable) in a cluster called 'a frame'. Each frame is designed for a particular MAC address which means an address of each network card. Within most of the cards, the address is invariable and allocated by the producer. Furthermore, in the same network segment, each card gets all frames passing through the transfer medium and finds out whether it is meant for it. If so, the card passes the frame on to a higher layer for processing. If not, the frame is ignored. All the same, a unique situation, in which a frame is designed for all MAC addresses, may arise (broadcast).

Let's consider that from a computer connected to a LAN network you are connected to a local mail server. Further, you enter a user name, password and you download your mail

without any problems. In fact, you are unaware of the fact that your colleague next to you or somebody on the opposite side of the Earth might be reading your password or mail. All the same, this situation is possible. The colleague might do it in a very easy way. As a matter of fact, there are programs allowing reading of frames even though they are meant for a different MAC address. These programs are called sniffers. You just need to install and start the program; the rest is performed by the program itself. Actually a skilled programmer finds writing of such a program very easy. The program switches the network card to so called "promiscuity" mode and monitors and analyses the entire traffic in the particular network segment. There are simple sniffers, e.g. Unix Tcpdump, or very sophisticated ones such as Hunt. The effective protection against Hunt is an adequate encryption. As long as the traffic is encrypted, the hacker will find the data useless since he would not be able to read them. The most advantageous alternatives are SSH and SSL. Therefore, it is convenient to replace current services using text passwords. Nevertheless, we need to know who is monitoring the traffic. On condition somebody from our local network is monitoring it, the adequate protection is quite hard to provide. All the same, there are programs which are able to find out network cards that are currently in the promiscuity mode.

All the same, sniffer may be installed to a computer within a certain LAN by an intruder from the outside. If he succeeds in hacking such a network, he will probably do so since there is no easier way of getting a great deal of information about various usernames and passwords. The most effective protection is to secure the network so that nobody would hack it and install the sniffer.

DNS

DNS (Domain Name Service) refers to one of the most important services that can be found within the network. This service secures transferring domain names to IP addresses and vice versa. Moreover, the service daily makes life easier to millions of users. Each object (server, router) which is a part of a network based on IP (intranet, internet) has an exclusive identification. Such identification is represented by IP address. It refers to a number in the form of xxx.xxx.xxx.xxx.; for instance, 192.168.1.1. This number is exclusive and cannot be shared by two objects connected to the network at the same moment. However, this definite identifier has its nominal alternative, e.g. http://www.firma.cz.; and namely the transfer between these two identifiers is secured by DNS. On the whole, addresses such as http://www.pcworld.cz or http://www.google.com are easier to remember than a "bizarre mixture" of numbers. All the same, a detailed description of DNS functioning is the aim of this article; therefore a closer look at DNS abuse will be taken in the following text.

DNS spoofing

DNS spoofing is a rather dangerous activity. In order to perform it, it is necessary to meet a few requirements. Above all, the hacker has to monitor your network traffic (see above). Provided he has access to your LAN, he can start a program on his computer which captures all DNS queries and tries to respond to them according to the hacker's intention. Thus, the key goal is to subvert the DNS response and redirect the guest to

another system. For instance, the hacker has a network server which is a perfect copy of, for example, a email web server - at the first sight, identical to the original. Its essential aim is to redirect all users of your network straight to that server. In this way, as long as a user enters http://www.webmail.something with the actual address 192.168.1.1 in his browser (this address is actually not available since it refers to an address from a private address block used for networks not directly connected to the internet; it is used just for an illustration), the hacker's program will try to subvert the DNS response and respond to the client in the way of the server having http://www.webmail.something 192.1691.100 address. On condition this response arrives in the user's system before the response of the actual and original DNS server, the hacker wins. How could such attacks be prevented? The first and general principle is to prevent the hacker from entering the network. Unless allowed to enter, he cannot do anything. However, if an authorised user behaves in such a way, a sniffer search engine is a comprehensible solution. In addition, there is another option - using a DNS server which signs its responses.

Further DNS abuse

There is one more type of attack which abuses a DNS server. However, this attack is rather old and effective only against older DNS versions of BIND server (Unix DNS server), still these old versions may be sometimes encountered. This attack is called Cache Poisoning and is based on subverting some authorised DNS responses. In order to pull this trick, the hacker does not even have to have access to your network; on the other hand, the solution is very simple - to use an updated software.

Routing and forwarding

Routing refers to the characteristic of IP protocol which manages the way of packets going through the network on condition that systems, which the hacker allocates on the way, allow routing. By means of routing, packets may be easily spoofed; thus, unauthorised information may be obtained. A comprehensive solution consists in switching off routing (neither Windows, nor Linux usually provide this service). Nevertheless, as long as you need routing, ACL must be correctly set in all routers.

On the other hand, routing allows a hacker to access to the internal network, which may be connected by one system and does not have to be visible to the surrounding internet on condition that forwarding rules are not adequately drawn up; therefore these rules should always be checked and tested.

Kidnapping sessions

Kidnapping sessions refers to a more advanced method of attack. The hacker has to have a sound knowledge of network communication as a result of which this attack is more dangerous. Again, the hacker has to monitor the network traffic. An ingenious device for kidnapping session is called Hunt. The hacker starts Hunt program on his computer and follows the communication between two systems. Upon considering a perfect moment for taking control over the session, he tries to take it over. In case of success, he is in a total control of the session and the target system will not find out that it is

communicating with a different subject. On the whole, the hacker is allowed to do everything like the original session owner. The adequate protection again consists in an appropriate encryption and heightened network security.

Man in the middle

These attacks concerns less sophisticated methods and are possible to be easily prevented. In fact, they strongly rely on user's indifference and carelessness. These attacks may also be carried out in case of encrypted protocols such as SSH or SSL. All the same, there is no error in protocols; only users' trust was abused.

SSH

As it appears from the title of this attack, the key principle is to provide a communication medium. The hacker needs to capture the client's session, connect to the actual target server and pretend to be the end-to-end system. In this way, the client does not know that he communicates via a third system. However, there is an intricate detail. In fact, each SSH server has its own identification key. Since the hacker does not usually have this key, he has to circumvent the security. In most cases, he creates his own key. However, it results in a situation that if a client connects to the hacker's system, a warning of a change of the key displays. On the whole, this attack relies on user's indifference and carelessness. Provided a user blindly obeys and ignores all warnings, the hacker is free to follow the traffic. Therefore, as long as you come across such a notice, contact the server administrator.

SSL

The attack by a medium may be also applied to SSL protocol. The principle is the same. However, this kind of attack is even more dangerous since SSL protocol is used within website, which associates lot of inexperienced users, who are, from image operating systems, used to blindly agreeing with every displayed notice (window). For that reason, it is very important to check SSL certificates whether they are signed by an official certification authority. A special attention should be mainly drawn to self-signed certificates.

Passwords

Even though your network is entirely secured, users observe security rules, everything is carefully monitored, controlled and analysed, yet there is a way of your network being attacked. Most servers offer certain services without which the internet would not live up to expectations. As a matter of fact, accessing these services may be offered via network and different groups of users. Some services may be available to everybody; others only to someone. Regardless of to whom the service is offered, the attack may be always carried out through the network. As a matter of fact, the hacker will try a lot of different methods and tricks; however, if proved that a network is utterly secured and most of common attacks failed, the hacker may attack your passwords. All the same, this attack is relatively time consuming and results are not guaranteed; nevertheless, it is the last possible method of getting into the system. Passwords of all protocols, which provide a particular authentication, may be attacked; e.g. SMTP, POP, FTP, SSH etc. This attack may be carried out by dozens of devices. Let's have a look at different methods of

guessing the password and how to prevent it.

Dictionary attack

Dictionary attack is the most frequent method of guessing passwords. It requires a fast processor, optimized program and long wordlist. By means of an adequate program, it is possible to attack almost all passwords. However, guessing passwords does not concern only attacks through the network; passwords could also be guessed locally, which depends on specific requirements and circumstances. The main principle consists in taking a word, editing it into an adequate form (in regard to the algorithm) and comparing it with the original password. Network services aims at sending this password over the network (its form is defined by the protocol). Dictionary attack chooses such words for potential passwords which are stored in a file. Thus, different sign combinations are not tried out. Some dictionaries are very comprehensive; therefore, it is convenient to choose such a password which is not contained in the particular dictionary.

Password creation

Try creating a random password and write it somewhere. Then we will do a test whether the password was carefully chosen. Now we need a dictionary. I recommend downloading dictionaries from http://www.phreak.org/html/wordlists.shtml address. It concerns a wide selection of dictionaries of various languages. You can download dictionaries according to topics or languages. I recommend downloading all. Now, we need to find out whether the chosen password is a part of some of them. For example, UNIX uses *grep mojeheslo slovnik*. Here are a few pieces of advice on creating a strong password.

Password should contain:

- Small letters of English alphabet a-z
- Capital letters of English alphabet A-Z
- Numerals 0-9
- Punctuation marks *, #, @,] etc.
- 6 characters minimum

Password should not contain:

- any word
- any name
- combination of a name (word) and numerals, e.g. honza52
- Information about you, nicknames or wife's name etc.
- numbers related to somebody or something
- Birthdates, phone numbers, addresses etc.

It is necessary to take into account that one password should not be used twice. As a matter of fact, on condition you set up several e-mail addresses, you should supply them with different passwords. Actually, it could happen that your password will be re-

vealed (e.g. by means of sniffer); thus a hacker would be allowed to access a lot of other systems.

14. Literature

Habraken, Joseph W. Průvodce úplného začátečníka pro Počítačové sítě : není zapotřebí žádných předchozích zkušeností!. 1. vyd. Praha : Grada, 2006. ISBN 80-247-1422-1.

STALLINGS, William. *Local and metropolitan area networks*. 6th ed. Upper Saddle River: Prentice Hall, 2000. xvi, 478 s. ISBN 0-13-012939-0.

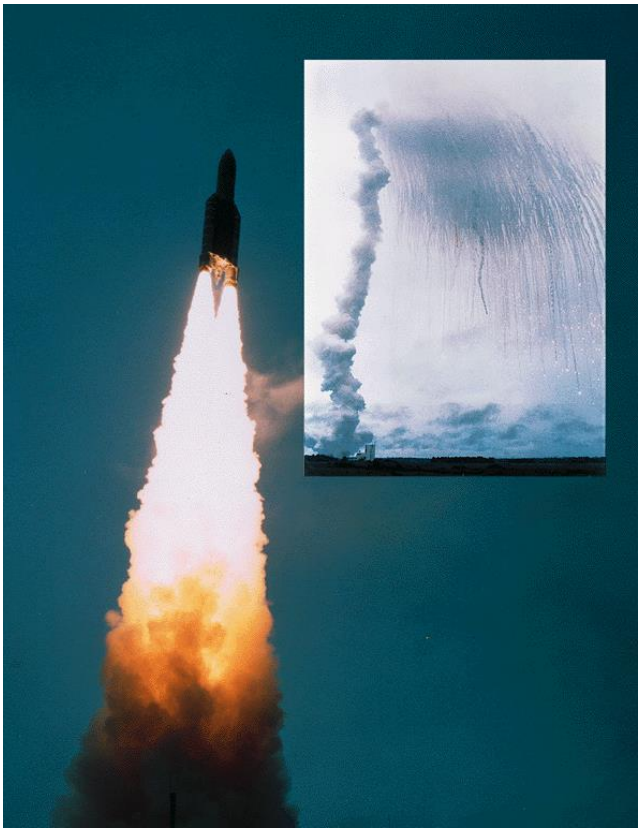
PUŽMANOVÁ, Rita. Moderní komunikační sítě od A do Z : [technologie pro datovou, hlasovou i multimediální komunikaci]. 2. aktualiz. vyd. Brno: Computer Press, 2006. 430 s. ISBN 8025112780.

THOMAS, Robert M. *Lokální počítačové sítě*. Vyd. 1. Praha: Computer Press, 1996. 277 s. ISBN 80-85896-45-1.

SOFTWARE ENGINEERING AND MODELING

1. Introduction to software engineering

Software error may cause either an application crash in a phone, or a really big problem. Some of the adverse consequences of faulty software are suggested in Richta's lecture (2011):



Fall of Ariane-5 rocket: As a result of an error, the main computer issued an order to deflect nozzles of solid rocket boosters and engine nozzles. Thus the rocket course was dramatically changed. In addition, given to aerodynamic forces, the upper part of the rocket broke off. Subsequently, the self-destruct system of the rocket was activated and the rocket turned into a cloud of burning shards. Total sum: 100 Mill. $ (including Cluster satellites, which were parts of the rocket, 500 Mill. $).

The North-east blackout of 2003 in the USA: It was caused by an unregistered blackout of the automated fault detector in the power station at the Niagara River. The blackout gradually spread via the network. The blackout decommissioned 21 power stations altogether. About 50 Mill. people were affected; three people died; damage amounted to 6 Bill. $

The collision of the landing module on Mars (1999): There was a problem of communication between components - the interface user expected a value in kilometres; however, the provider reported it in miles. Instead of planned 140-150 kilometres above the surface, it landed 57 kilometres above the surface. In such an altitude, Mars atmosphere is too heavy for a probe. Climate Orbiter probably burned in the altitude of 80 kilometres. (Source: Technet.cz)

1.1. Reasons for software engineering development

At the beginning of computer development in 1940s and 1950s, computer computing power was very low. In time, this computing power was developed by geometric series according to well-established Moore's law. Software development was still in its infancy. No methods or techniques which would facilitate the development were devised. In 1960s, when computing power significantly developed again, so called software crisis arose.

The distinctive features of the software crisis were unsustainable delay and price-increase of projects, poor program quality, maintenance and innovation difficulties, low efficiency of programmers, development ineffectiveness, poor results etc.

Main causes of the crisis are as follows:

- Poor communication between people involved in software development and between developers and customers.
- Wrong approach. The particular software was developed and approved by a developer and a dissatisfied customer was referred to as a person who does not understand the matter.
- Bad planning of the project. Developers hoped to eventually meet the deadline in a way.
- Unrealistic estimates of the duration of development, costs and scope.
- Low work efficiency. Programmers paid attention to unimportant things and ignored the essentials.
- Ignorance of basic rules. E.g. Brook's law from 1975: "Adding a research capacity to a delayed project will only increase its delay."
- Underestimation of threats and risks. The threats were not regarded; instead of being prevented, they caused big troubles.
- Bad technology application. False idea that all problems will be solved after introducing a new technology.

As a response to the above-mentioned issues, a conference which would introduce basic principles of software engineering took place in 1969.

At the time, it was defined as follows:
"Software engineering is a discipline which deals with establishing and applying key engineering principles to software development in order for us to achieve economic software development which is reliable and works efficiently on available computing devices."

Software engineering can be referred to as an engineering discipline dealing with actual problems of development of complex software systems. This discipline involves several complex thought processes. This engineering discipline follows pragmatic approaches. These approaches include knowledge of special requirements for the software product, its analysis, design, implementation, testing and installation of the system to the end user. At the same time, they involve managerial approach to the project management,

which enables effective use of individual techniques whose main goal is to effectively devise a high-quality software product.

1.2. Psychological excursion

Cognitive psychology explained different ways in problem solving between experts and novices. Experts are able to recall partial solutions of similar situations and apply the previous knowledge to the identification and definition of the problem (Eysenck and Keane, 2008). Experts deal with the problem by exploring the heart of the matter and all external circumstances. Based on this information, they try to find a solution. Novices, on the other hand, try to guess a solution and afterwards assess whether it is possible to work out the suggested solution from relevant facts. They thereby use the backward shift strategy (Nolen Hoeksema at al., 2012). According to Plháková, the biggest difference between novices and experts lies in the extent of knowledge organization (Plháková, 2003). Highly developed self-regulating abilities (explored in a separate chapter) comprise characteristics of successful individuals (experts). These abilities allow effective use of knowledge and skills (Helus & Pavelková, 1992), In fact, laymen knowledge to understand problems is usually very superficial. On the other hand, experts first approach the heart of the problem and try to identify and organize it. They do not approach the solution until they have a feasible plan. Furthermore, experts focus on preparation much longer that laymen, but find the optimal solution to the problem much faster (Plháková, 2003). Říčan (2016) says that experts: "Not only do they have greater knowledge (quantitative aspect), but also differ in a qualitative way; i.e. in terms of a more developed strategic behaviour when dealing with problems (they thereby differ in their schemes for learning). Experts have better organized domain-specific knowledge; therefore, they are able to get new information from their field of study faster since the previous knowledge, which is available, functions as an integrating structure for new incoming information via associative memory with only a minimal cognitive effort."

In general, we can say that software engineering integrates problem solving strategies which, according to cognitive psychology, only experts are capable of.

2. Life cycle of information systems

Wikipedia aptly likens the life cycle to the bridge. The bridge is actually a means for making human activities easier. Accordingly, software systems should serve the similar purpose. Like bridges, software has to be designed, used, maintained until the end of its lifespan and eventually decommissioned. In construction engineering, there is a life cycle of particular buildings which is very similar to that of software.

There are a lot of classifications of software life cycles. We have chosen Systems Development Life Cycle (SDLC) by The Department of Justice of the USA from 2003.

- **Initiation** - firstly, it is necessary for the object with enough money (customer) to feel the need of an improvement implemented by a software system. Let's call it 'intention'.
- **System Concept Development** - this intention is re-examined in terms of feasibility and suitability. The system scope, on which base costs that must be approved by the customer are calculated, is loosely defined.
- **Planning Phase** - this concept is developed in a way that describes the function of the enterprise after the installation of the approved system. In addition, specific project plans are drawn up and approved.
- **Requirements Analysis Phase** - all requirements are profoundly set out, which enables a further development of the system. These requirements usually concern data, computing power, security system, requirements for maintenance etc.
- **Design Phase** - physical features of the system are designed within this phase. Main parts of the system, its inputs and outputs are defined. All that requires an input or user's approval has to be recorded and re-examined by the user.
- **Development Phase -** all specifications laid down in previous phases are put in the developed software. They are subsequently tested.
- **Integration and Test Phase** - individual system components are integrated and systematically tested.
- **Implementation Phase** - the system is installed and launched at the customer. The customer must test and approve of the system. The phase ends by putting the product into operation.
- **Operations and Maintenance Phase** - The operation of the system should be adequately monitored. If necessary, required system modifications are introduced. This process goes on until the system can be effectively modified to the customer's requirements. If the process is not possible to continue or it is too expensive, a phase of a new software planning begins.
- **Disposition Phase** - decommission of the system. A special attention is paid to

precisely storing data processed by the system in order for the information to be effectively transferred to another system or archived in accordance with official regulations and policies of the record administration in case of a future access.

The entire software life cycle is described in the following text. Individual methods of software development will be shown in order to closely focus on RUP method (Rational Unified Process). This method involves phases from Initiation to Implementation. RUP method works a lot with UML; therefore, six different kinds of UML diagrams will be shown. Business Process Model and Notation, which can be useful at the beginning of new software development, will be discussed. Software testing and its maintenance by means of ITIL method will be described in a separate chapter.

3. Software development methods

In general, they refer to procedures, rules and devices leading to the software development. Sometimes they are also called software process. Individual devices designed for software development are called Framework. The aim of these procedures is an effective development and maintenance of the software.

Following these procedures, we reduce the risk of unexpected problems and make the work schedule of particular software clearer. If individual developers follow the same procedures, they thereby encourage the cooperation thanks to the previously formulated program development rules.

As time went on, a lot of software development methods were devised. All the same, until now, there is no detailed and clearly defined frame of a software development method which could be considered as referential. These methods include:

3.1. Waterfall model

Waterfall model is a sequential and linear process which is approached as a sequential progress which flows through individual phases of conception: design, testing, integration, maintenance and alternatively transition. This model originated in 1970s. It can be regarded as an essential model on which other models are based.

Moving from one phase to another should be carried out only when its preceding phase is partially or completely verified. Unfortunately, this model is often not possible to be made use of in real life. For example, the customer often does not change program specifications until the stage of program development or until its application. The main problems are as follows:

- It is not possible to assess the final quality of the product in the course of software development on the grounds of previously established criteria for software specifications.
- The final product depends on the initial requirements for the final software.
- The time needed for the software development is too long.

In regard to this model, an effort to remedy defects would result in development of other models. These processes are explores by prof. Vondrák (2002):

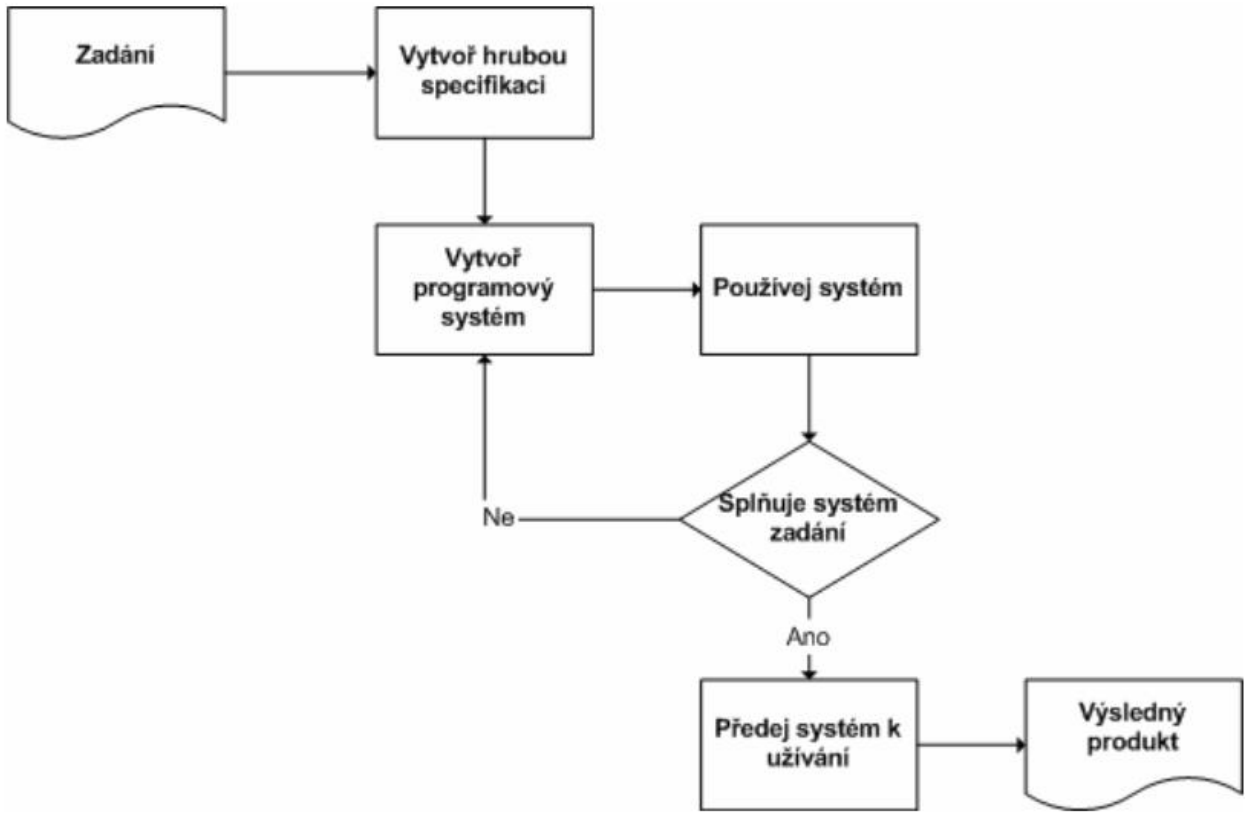
3.2. Iterative and incremental development

Individual steps are taken in repeated cycles “iterations”. Each iteration takes on a smaller portion (increment) of functions into the developing software which leads to the target state. Thus the required program is being developed and its functional capabilities

are being enhanced. The customer thereby has the opportunity to try out the developing software as soon as its early phase of development. In this way, he can specify his requirements for the developed software more effectively. Thus errors, defects or wrongly imposed requirements for the program will be detected significantly earlier. Owing to the fact that the customer sees the software from its early phases, he can be more involved in the final form of the product. In contrast to Waterfall model, differences between individual phases are very slight. It is rather a parallel in which iteration represents one small waterfall. RUP process, which will be dealt with in a separate chapter, also falls into this category.

Exploratory programming

Prof. Vondrák (2002) regards this model as a deterrent example which is still sometimes used. He calls this model Exploratory Programming.



3.3. Agile software development

Most of the software processes originated in large corporations. These processes strictly required following the scheduled procedure and obligatory creating a lot of documents. That is fine in large companies where a lot of people work on the project, or where a high level of reliability is required (e.g. power grid control). In case of small projects, the time spent by obediently following all the procedures and other formalities was much longer than the time meant for programming and testing.

In response, agile methods or techniques arose at the beginning of new millennium. Basic principles of these techniques were formulated in the Manifesto for Agile Software Development.

- Individuals and interactions are more important than processes and tools.
- Working software is more important than comprehensive documentation.
- Customer collaboration is more important than contract negotiation.
- Responding to changes is more important than following a plan.

The basic principle of these methods is to provide developers with the environment and support they need for the software development. Developers focus on the software development, its design; they do not excessively deal with documentation. These methods fall into iterative methods. Its high priority is active customer collaboration. Work schedule is usually drawn up within the next iteration. Because of their "anarchistic" nature, these techniques are suitable for smaller quality developer companies. The key principle of conveying information within a development team is face-to-face communication.

Currently, there are several agile methods Such as Extreme programming and Scrum (Sklenář, 2007).

4. RUP

RUP process (Rational Unified Process) originated in the second half of 1990s in collaboration with several companies under the leadership of Rational Company. This company became an IBM division in 2003. RUP is not the only established process; it is rather an iterative process framework which should be adapted to the needs of developer organizations and a software development team. This adaptation is implemented in the way that development teams chose process elements which correspond to their needs. RUP originates from UP (Unified Process). RUP currently belong among widespread software process models. A lot of tools support it.

A common Waterfall model defines separate roles. It thereby happens that an analyst accepts customer's idea of the system functioning. This idea is subsequently translated to a document, which will be handed over to the programmer. If there is no effective collaboration between the analyst and developer, the developer has to focus on requirements set out in the document. In such a case, developer's interpretation of the requirements can significantly differ from customer's original idea.

This process is built on several principles. This process in based on the idea of **software iterative development**. Thanks to that each program should be developed in recursions (iterations). Iteration should be created by an executable code.

Žáček (2017) lays down basic RUP principles:

- The effort to minimize project risks as soon as possible on a regular basis.
- To make sure that customers are provided with added value.
- Focusing on executable software.
- To undergo changes in early phases of the project.
- Early drafting of executable architecture.
- Re-using existing components.
- Close collaboration - all are one team.
- Project quality is determined from all its proportions, not only a part (testing).

Original requirements for the software are essential for the quality development of the product. RUP creates **Requirement Management System**. It manages their acquisition, documentation and monitors changes in requirements. Subsequently, this system is used by developers in communication with customers. Changes in the software development are managed in the same way.

A lot of parts of the software product are alike. Therefore, their reinvention is waste of time. RUP introduces **Component-Based Development**. As soon as we have necessary components, the development of the product starts to integrate relative components. This integration can be likened to Lego or building up a desktop computer.

Thanks to the complexity of developed programs, **software visualization** is very important for a better idea. RUP works on UML language.

Verification of software quality is a very important aspect for the feedback to developers

in order to satisfy customer's needs thanks to the working software. This verification should involve all product developers. RUP specifies specific procedures assessing the software quality.

RUP software development is divided into four phases. RUP phases could be referred to as individual project stages and its changes in time. Each phase often consists of several iterations.

4.1. Schematic model

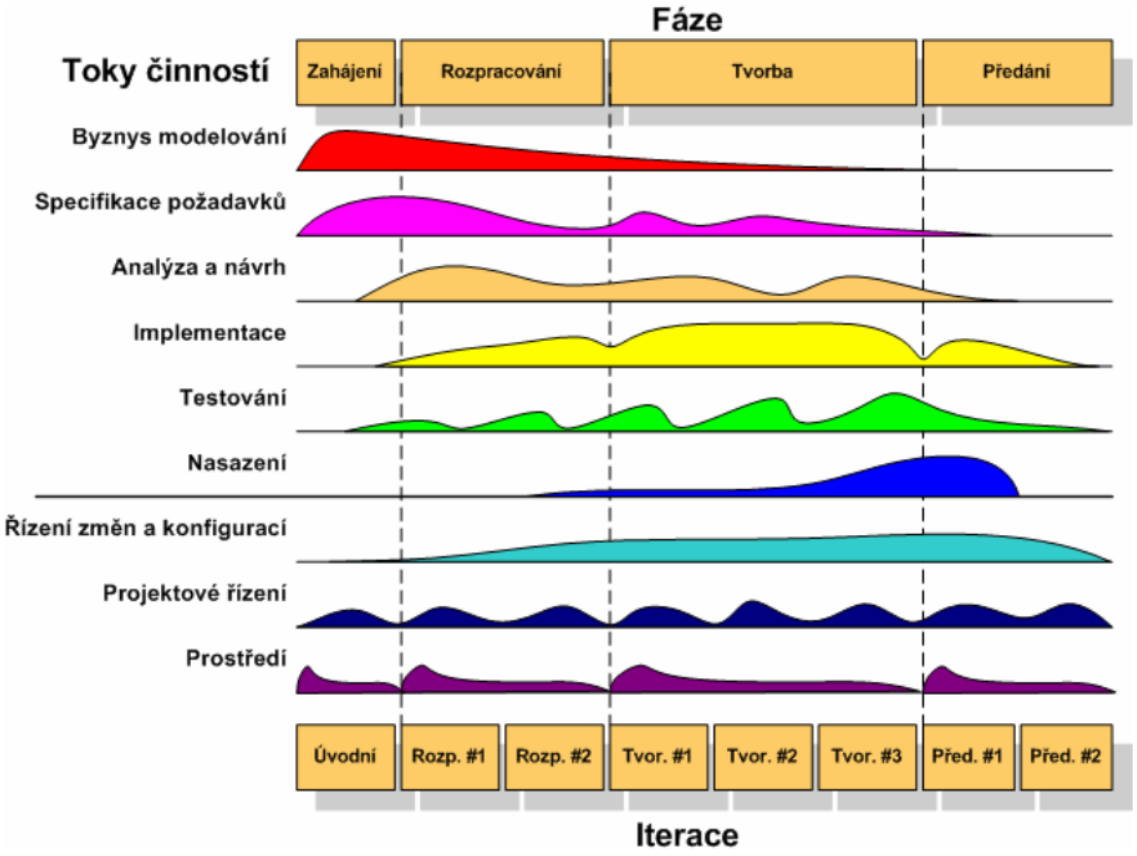
As such, schematic model consists of six engineering disciplines and three auxiliary disciplines. Engineering disciplines include:

- Business process modelling
- Requirement specification
- Design and analysis
- Implementation
- Testing
- Deployment

Auxiliary disciplines involve:

- Configuration and change management
- Project management
- Preparation of the project environment

A relative frequency of activities of individual disciplines in correlation with individual phases and project iterations was worked out by prof. Vondrák (2002).



A relative time and source distribution should be as follows:

	Inception	Elaboration	Construction	Transition
Sources	Ca 5%	20%	65%	10%
Time	10%	30%	50%	10%

Žáček (2017) suggests outputs of individual phases:

- The output of Inception is the understanding of crucial issues, the aim of the project, assessed risks.
- The output of Elaboration is executable and verified architecture (=
- Functioning part of the application).
- The output of Construction is beta-release application - a relatively stable, re-executable and almost completes application.
- The output of Transition is a product ready for final deployment including the entire documentation and hardware. The major difference also lies in the fact that each phase is characterized as follows:

Individual phases involve various numbers of employees and posts. It is important for the employees to always work together in one or more teams. If needed, they help and advise each other.

4.2. Inception

At the beginning, it is necessary to assess the project size, the requirements, time and financial costs of the whole project implementation. Žáček (2017) suggests 5 goals of this phase:

- Understanding the process - conveying the vision, defining the system size, its limits; understanding the person who wants the system and how he could benefit from it.
- Identification of key system functionalities - identification of the most critical Use Cases.
- Suggestion of at least one possible solution (architecture), (i.e. Whether it is possible to continue in the development Note: author)
- To be informed about costs, project conception and risks.
- Definition/modification of the process; tool selection and setting.

Definition and imposing of requirements
Within imposed requirements, we try to obtain maximum information about what users expect from the new system and which essential functions the system is supposed to perform. Techniques of data collection include:

- Documents and records

- Interviews with future users
- Questionnaires
- observation
- Oral history of users

These requirements can be divided into:

- **Functional requirements** these are requirements for system functionality from the system; alternatively, system functional capability and incapability should be defined. A use case diagram is an effective tool for illustration of functional requirements.
- **Non-functional requirements** these are requirements for system properties as a whole. They can concern efficiency, reliability, security, system capacity; alternatively, they can refer to organizational requirements such as maintaining standards and following established procedures. Alternatively, they can involve requirements regarding valid legislation.

Goals

The key goal is to convey a vision of the project. This vision should be defined in regard to the user. However, it should at the same time contain a basic technical view of the applied technology or architecture components. This vision should mainly set out requirements for the future software. This phase takes place only once. All the same, in more complex cases, it can be repeated several times.

At the same time, the degree of risks should be assessed in certain areas; these risks are as follows:

- Risks of technical branches (technologies, compatibility, communication with other systems)
- Financial risks - development costs and subsequent maintenance
- Time risks - the more sources we have, the shorter the independent development is likely to be

At the end of this phase, it is necessary to assess whether the development is possible to continue. The sooner we put an end to an obscure project development, the lower the costs of the project development are. Lifecycle Objective Milestone (LOM) is used for that purpose with evaluation criteria as follows:

- Interested parties concurrence on the project scope, costs and schedule estimates
- Agreement that the right set of requirements for the system have been imposed and that there is a shared understanding of these requirements
- Agreement that the cost and schedule estimates, priorities and risks correspond to the development process
- All risks have been identified and mitigation strategies have been decided on

4.3. Elaboration

This phase initiates forming of the project. The aim of this phase is to establish the system architecture in order to produce a majority of healthy functionalities in the next phase. This architecture is based on findings from the previous phase. Crucial issues of the architecture design of the developed software are to be dealt with here. The system functionality does need to be perfect. Nevertheless, the majority of essential functionalities should have already been integrated into the system. The research team should be aware of the fact that the architecture design must stabilize during this phase.

At this point, the crucial issue is analysed and the project architecture acquires its basic form. The accurate architecture design efforts to mitigate risks associated with the software development. These risks mainly concern:

- Time and finance
- Architecture accuracy
- Processes and technologies
- Software development objective

A series of small iterations can mitigate the degree of above-mentioned risks at this point; they also help monitor the principal objective of the development. If we create the system using similar technologies like before, there is obviously a lower degree of potential risks. Therefore, we are able to reach goals of this phase within iteration. On the other hand, if we are not familiar with the used technology or the project is more complex (e.g. Stricter safety requirements), we need two or more iterations. In case this phase sees a more substantial change in the architecture, another iteration, which will verify necessary functionality and stability of the modified architecture, should be added. A basic principle "the later the foundation of the construction is changed, the more expensive the whole reconstruction will be" applies here.

In order to verify the software development objective, a beta version of user interface, on which the most important system functionalities would be tested, should be created.

At the end of this phase, the system architecture must be stabilized. The executable architecture must verify the system architecture and simultaneously serves as a means for verifying critical system functionalities. These critical functionalities must also be modelled, for example, by means of UML sequence diagram. We also try to find identify potential problems and risks. We should afterwards group selected objects to packets regarding visibility rules and future product configuration. We should also have an idea how the data in the database will be handled, as a result of which we should regularly integrate the components. That means to determine the order and way in which the selected components will be integrated.

Eventually, testing of critical scenarios constitutes a very important part of this phase. Testing of critical scenarios may demonstrate our progress in the elaboration phase. Apart from that, critical scenarios should be tested in regard to aspects such as heavy system load, sufficient system architecture performance or collaboration with external

systems.

At this point, we test system objectives by means of Lifecycle Architecture Milestone (LCA), while LCA evaluation criteria are:

- The product vision and requirements are stable.
- The architecture is stable.
- The key approaches and procedures are tested and verified, so that they have been proven applicable.
- Testing and evaluation of executable prototypes have demonstrated major project risk elements, which have been successfully resolved.
- The iteration plans for the following phase are drawn up to allow the work to proceed.
- The iteration plans are supported by credible estimates.
- All participants agree that the current vision can be met if the current plan is executed in context of the current architecture.
- Actual costs versus planned costs are acceptable.

The project may be aborted or otherwise re-thought if it fails to reach the milestone.

4.4. Construction

The key goal of this phase is to build a credibly working software system while minimizing its costs.

This phase focuses on developing components and other system elements. This phase is further characterized by creating encryption which is heavily time and human (it requires a lot of programmers and testers) consuming. The rest of the functionalities are being designed here; i.e. remaining main ones (required by the customer) and most of the others. If an intervention in the system architecture is to be dealt with, the previous phase is likely to be wrongly completed.

Essential prerequisites for a successful completion of this phase are: architecture compactness, parallel development of individual teams, configuration and change management and automated testing.

Larger projects may perform several iterations (commonly 2 - 4). This phase usually performs the most of the iterations. Iteration planning corresponds to the number and type of not yet implemented functionalities. Each iteration deals with a separate project segment. Ideally, the architecture, which is further developed, used, or re-used, is created from previous phases. The system is subjected to integration and testing. With respect to the effective organization, one team is responsible for the architecture while other teams parallel work on the subsystem development. These “parallel teams” mainly communicate with the team responsible for the software architecture.

Thanks to the iterative development, a lot of files and their versions, which are subsequently tested and integrated, are developed. For that reason, configuration and change management must be introduced. This management registers individual configuration and changes. If such a system works, developers can entirely focus on the

further development and thereby enhance their effectiveness.

At the end of this phase, a beta version of this program must be developed including helps in the application, installation instructions, user manuals and tutorials. In this way, future users will be able to try out the software and give us a constructive feedback.

At the end of this phase, we should pose ourselves several questions according to IOC Milestone (Initial Operational Capability)

- The beta version is ready to be released to the first users (testers)
- Users are prepared and able to use our beta version
- Actual costs versus planned costs are acceptable

4.5. Transition

This phase aims at the system transition from its development to its regular use and remedying its major defects (they usually concern its performance, user-friendliness and functionality). These phase activities include:

- Training of administrators and end users
- System testing to verify expectation fulfilment of end users
- Preparation of marketing materials
- Environment and data preparation, for example data migration from the old to the new system
- The product is closely monitored in terms of qualitative frameworks established at the beginning of the project.
- Internal project evaluation, learning from mistakes and problems arising from working on the project.

English wikipedia from 24. 6. 18 further states: *"The system also undergoes Evaluation phase; each developer who does not carry out a purposeful work is replaced, or removed.*

This phase should also be finished by a milestone. PRM (Product Release Milestone) criteria are:

- Users are satisfied.
- Final costs versus planned costs are acceptable. Things to be changed in order to approach the problem

5. Business Process Model and Notation

The key goal was to devise a graphic tool which is comprehensible to all business users (from analysts through developers to business process owners), which means a more effective communication between the developer and customer. Business Process Model and Notation (hereinafter BPMN) consists of a set of simple graphic elements from which a business process model can be modelled. BPMN has been developed since 2005. Version 2.0. came into existence in 2011. Process flowchart is based on a flowchart which is very similar to UML activity diagram, which will be discussed in the further text.

Flow Objects
It is closely connected to information flow in the process. Flow objects are the most important graphic elements. They can be divided into three groups - Events, Activities and Gateways.

Events
Events are marked with a circle in which an icon can be displayed. It marks an event which directly controls the process flow. Events are divided into **Start** (Start event) **events**, which initiate the process and are marked with a circle with a narrow border - sometimes with green colour.

Furthermore, there are **End events**, which indicate the end of the process, or the result of the activity. These are marked with a bold border, or a bold icon inside the circle - sometimes with red colour. Some sources also mention Intermediate event which is marked with a double border.

Activity
Activities are marked with a rounded-corner rectangle. This element describes work or activity. The activity can be either **atomic** (i.e. Task), which is considered a single unit and which must always be performed as a whole.

Alternatively, it can be broken down to a single process which is called Sub-process. This kind of activity is used in processes which we do not want to reveal on a particular level. Sub-process activities are marked with a plus sign at the bottom line of the rounded-corner rectangle.

Gateway
Gateways are marked with a square or diamond shape. It determines forking and merging of process flows, e.g. decision-making or parallel processing.

Connecting Objects
They connect flow objects or artefacts. Thanks to the connection, a new structure (framework) of the business process arises. Connecting objects are divided into three basic types:

- **Sequence Flow** - is marked with a solid line and arrowhead. It shows the order in

which the individual flowchart units should be performed.

- **Message Flow** - is marked with a dashed line and open arrowhead. It displays the message flow across organizational boundaries of the process.
- **Association** - is marked with a dotted line. It associates objects to additional information. It is also used for displaying inputs and outputs.

Swim-lanes

They display participants of the process and refer to the organization and categorization of activities in the process. They consist of two types.

- **Pool** - represents participants of the process, or alternatively, it separates different organizations. It can contain more lanes. One pool contains one separate process.
- **Lane** - is situated under the pool. It is used for organizing and categorizing activities.

Artifacts

They bring some new information into the process. They are independent from flows.

- **Data Object** - is marked with a bent-corner rectangle (a sheet of paper). It shows data produced in an activity; alternatively it tells us which data are required for an activity.
- **Group** - is marked with a rounded-corner rectangle and dashed lines. It is used to group different objects.
- **Annotation** - gives the reader a clear and understandable impression.

We provide a model of ordering and delivering pizza as an example.

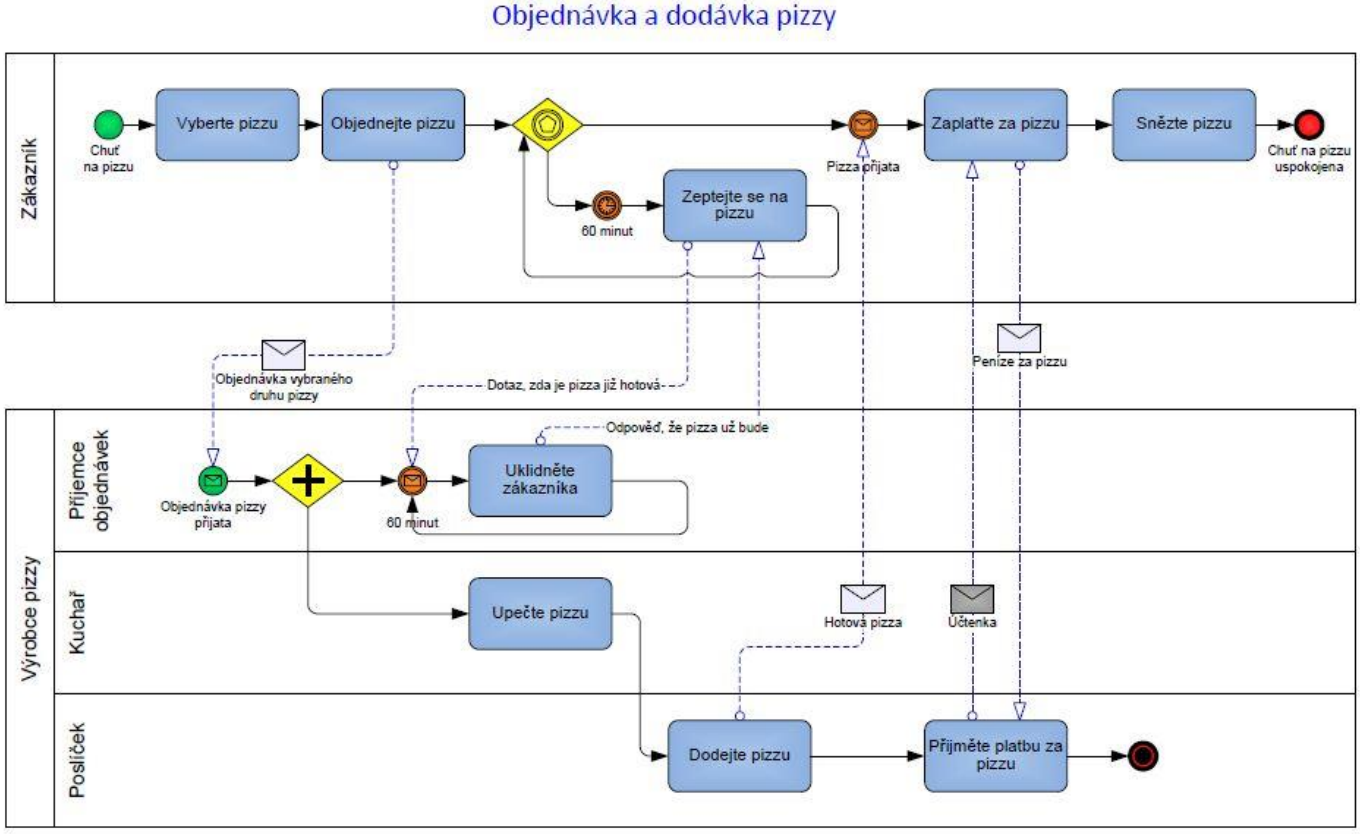


Figure 1 source: http://bpmn.horcica.cz/wp-content/uploads/2011/11/BPMN-2.0-Examples_CZ.-Dod%C3%A1vka-pizzy.jpg

6. UML

As a matter of fact, a lot of software systems are exceptionally complex. Thus, their development lays down stringent requirements for developers' imagination. When a project involves more people, the future software system needs to be precisely defined - modelled. There are a lot of approaches towards such modelling. All of them have their pros and cons. In order to encourage an effective cooperation, it is important for everyone to know, use and understand the modelling system its terminology. For that reason, UML (Unified Modelling Language) was developed. Standard UML is defined by Object Management Group (OMG).

UML refers to a unified language for drawing diagrams. UML enables specifications (structure and model), visualization (flowcharts), construction (Software development methods) and documentation of software system artifacts. There is a large scale of diagrams. Each of these diagrams is used for a different goal and in different project phases. UML provides 13 types of flowcharts in total. In this text, diagrams will be divided into Structure diagrams and Class diagrams.

Structure diagrams
Structure diagrams describe the project structure. They are often used to describe the architecture of the developed software. We will deal with several selected diagrams belonging to this category.

Class diagram
This diagram shows project class divisions, their relationships, operations and methods or methods of the methods. These classes are written in a rectangular vertically divided into 4 parts. This diagram is the cornerstone of object-oriented modelling. It is also used for data modelling. Furthermore, the diagram depicts a static structure of the system, and to some extent depends on a specific programming language. The aim of this activity is to suggest the way in which the product is implemented in the implementation phase.

Examples of class diagrams are provided below.

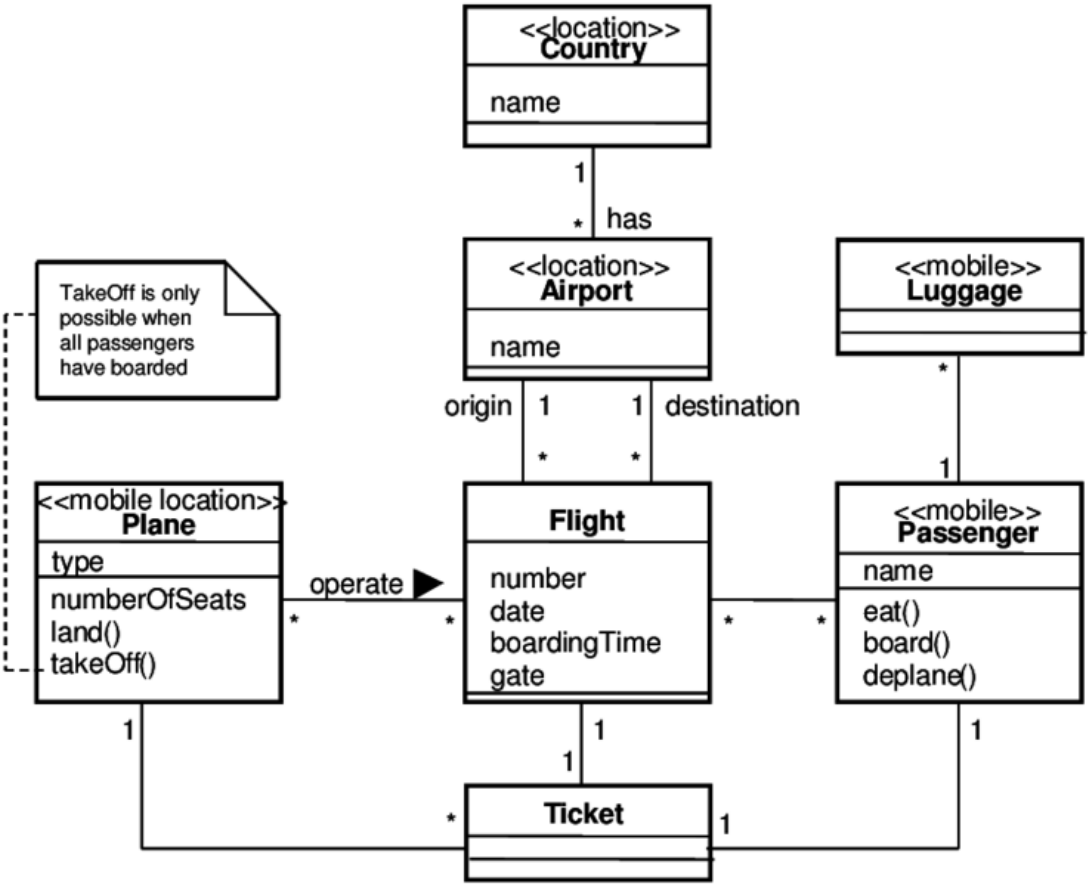


Figure 2 Source: Andrade et al, Recent Trends in Algebraic Development Techniques—16th International Workshop, WADT 2002, Frauenchiemsee, Germany. Vol. 2755 of LNCS.

Component diagram

This diagram works with components used in the system and describes the way the components are interconnected to form larger components of software systems. It is used to describe the structure of different complex systems. The diagram is mainly used for designing software by components. It also has a higher level of abstraction than class diagrams. One or more classes are implemented within a single component.

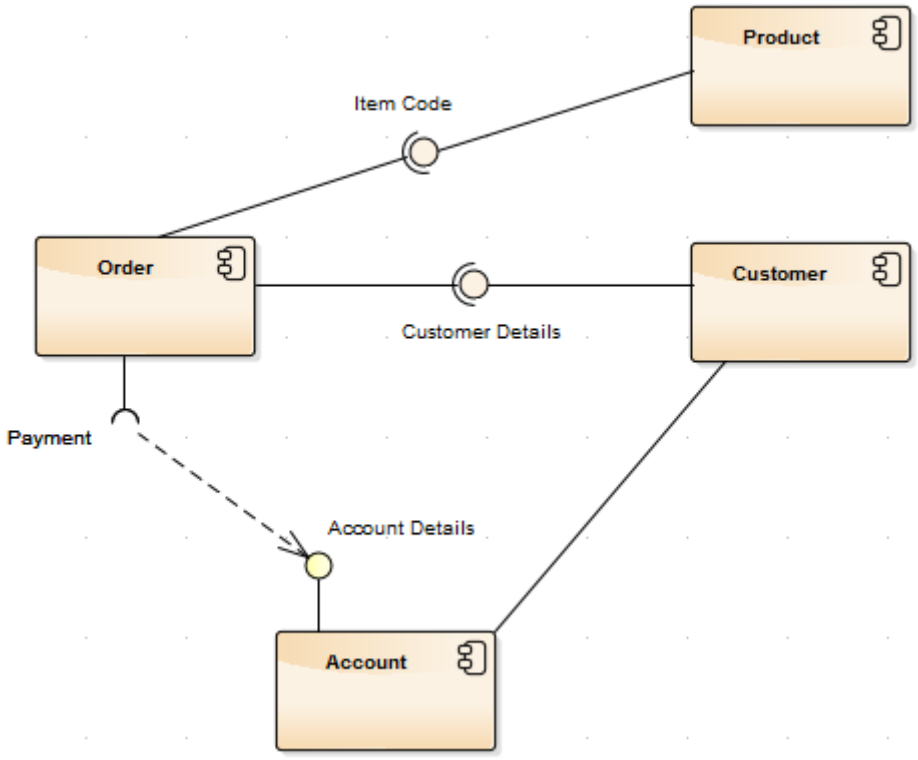
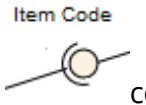


Figure 3 Source: http://sparxsystems.com/enterprise_architect_user_guide/13.0/model_domains/componentdiagram.html

The picture illustrates 4 components in total.



Assembly connectors connect the order interface with the product and the customer. In this case, components require ID product information and details about the customer.

A simple line between Customer component and Customer Account component illustrates the relationship between the components.

Object diagram

Object diagram depicts objects (class instances) and their relationships (links - association instances) in one moment. This diagram looks like a class diagram and can be actually considered as its specific example. It is mainly used when we want to show examples of data structures in one particular moment.

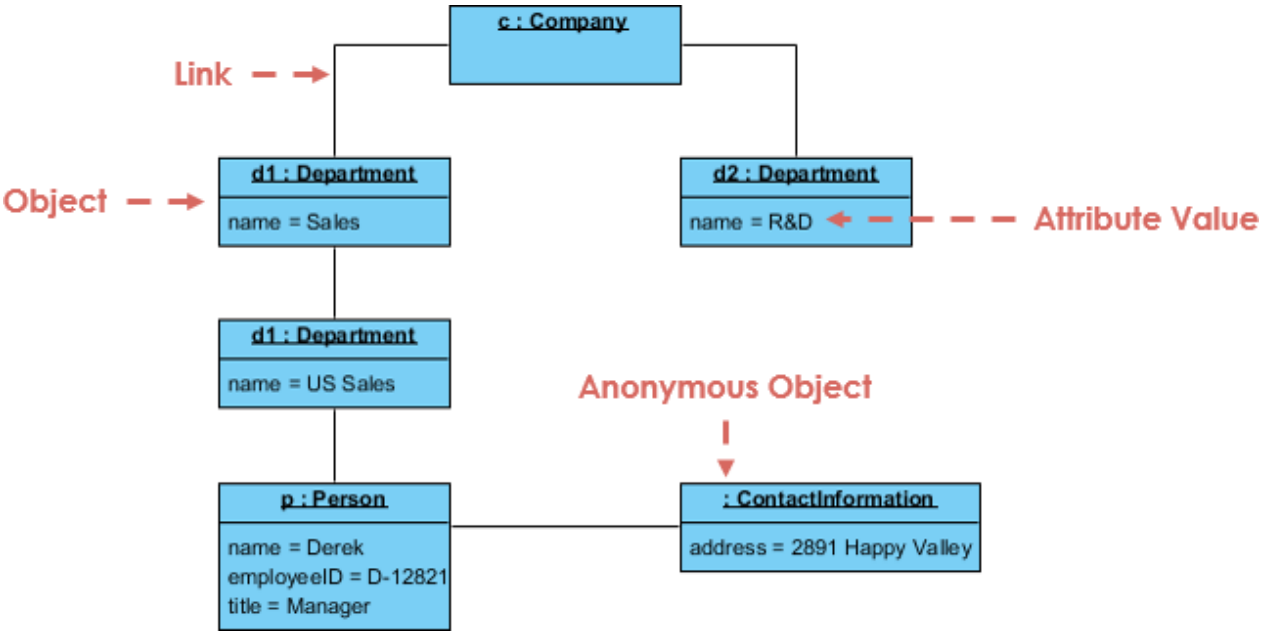


Figure 4 Source: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-object-diagram/

7. Behaviour diagrams

These diagrams describe separate processes by means of activities representing their states and their transitions. Regarding the fact that activity diagrams describe behaviour of the system, behaviour diagrams are used to describe software system functionalities.

Activity diagram

This diagram is designed for modelling computing and organizational processes (i.e. Work process) and can be used for modelling data flows. The diagram models processes by means of activities representing its states and transitions between individual states. Thanks to that it is one of the diagrams capable of identifying the behaviour of the system.

- Rounded rectangles represent actions.
- Diamonds represent decisions.
- Bars represent separate activities.
- The black circle represents the start (initial node).
- The encircled black circle represents the end (final node).

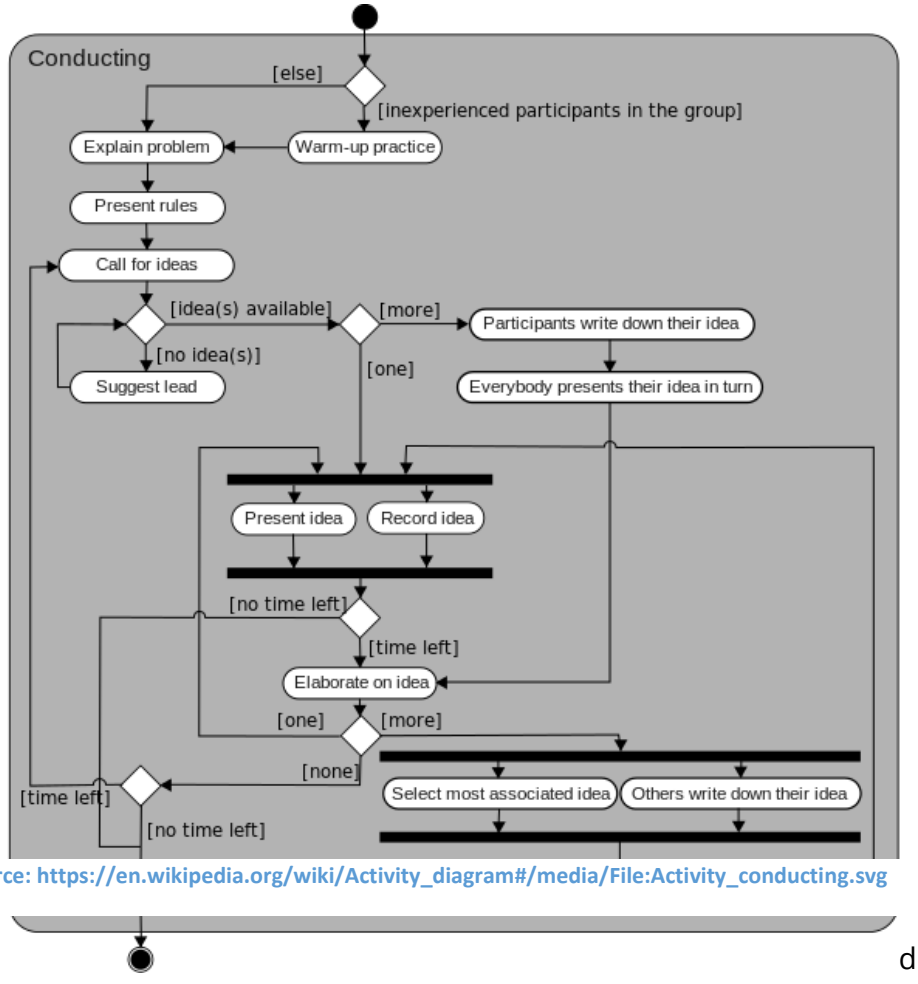


Figure 5 source: https://en.wikipedia.org/wiki/Activity_diagram#/media/File:Activity_conducting.svg

Sequence diagrams

Sequence diagrams displays object interactions which participate in particular system

functionality. Apart from object and classes, it displays sequence of messages necessary for achieving the particular functionality between individual objects in regard to their time-order. These diagrams are often used in the implementation of use cases regarding the logic of the development system. The aim of this activity is to suggest the way in which the product will be implemented in the implementation phase.

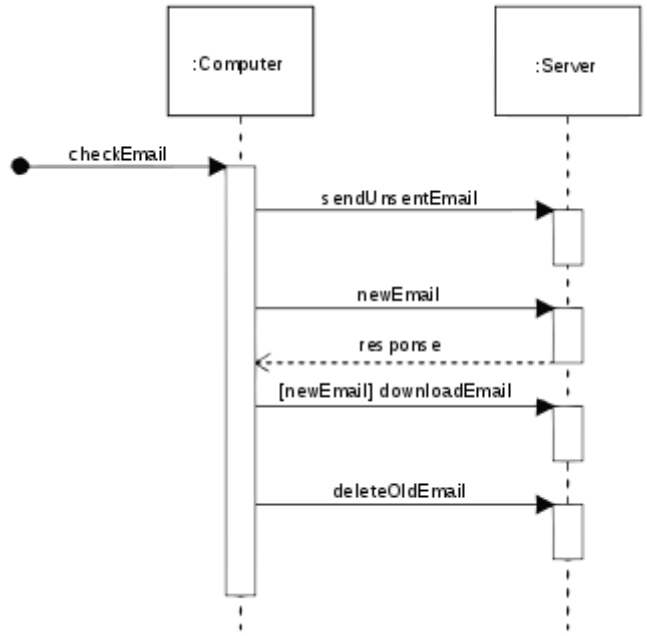


Figure 6 Source: https://en.wikipedia.org/wiki/Sequence_diagram#/media/File:CheckEmail.svg

The time-line in the diagram is vertical (time flows from the top down). The placement of objects is horizontal. Our picture depicts 2 objects (Server and computer); dashed vertical line is also called the lifeline. Our case refers to objects which had existed before the diagram was drawn and will exist even after. Rectangles on the lifeline illustrate an activity. Horizontal arrows represent messages. The arrow direction displays the sender and addressee. The full line demonstrates the mandatory message, while dashed line represents a non-mandatory one.

Use case diagram

It usually represents the relationship between the customer and the system, where it demonstrates different use cases in the customer-system relationship, which is often verbally described by means of scenarios. In order to map scenarios to use case diagrams, use case implementations are applied. Here, the algorithm of transcription of scenario entries to the diagram is to be dealt with.

For example, BPMN can be the first impulse for this diagram to be drawn. The result of this is a list of activities which can be performed by the developing software instead of, for example, people. Diagram figures are called actors. Actors are associated with relevant use cases. The simple line is called association. The horizontal direction further demonstrates the structure of the actors. The bottom actor inherits characteristics from the actors above him. This type of relationship is called generalization. Include relationship is implemented when the use case from which this relationship develops is imple-

mented.

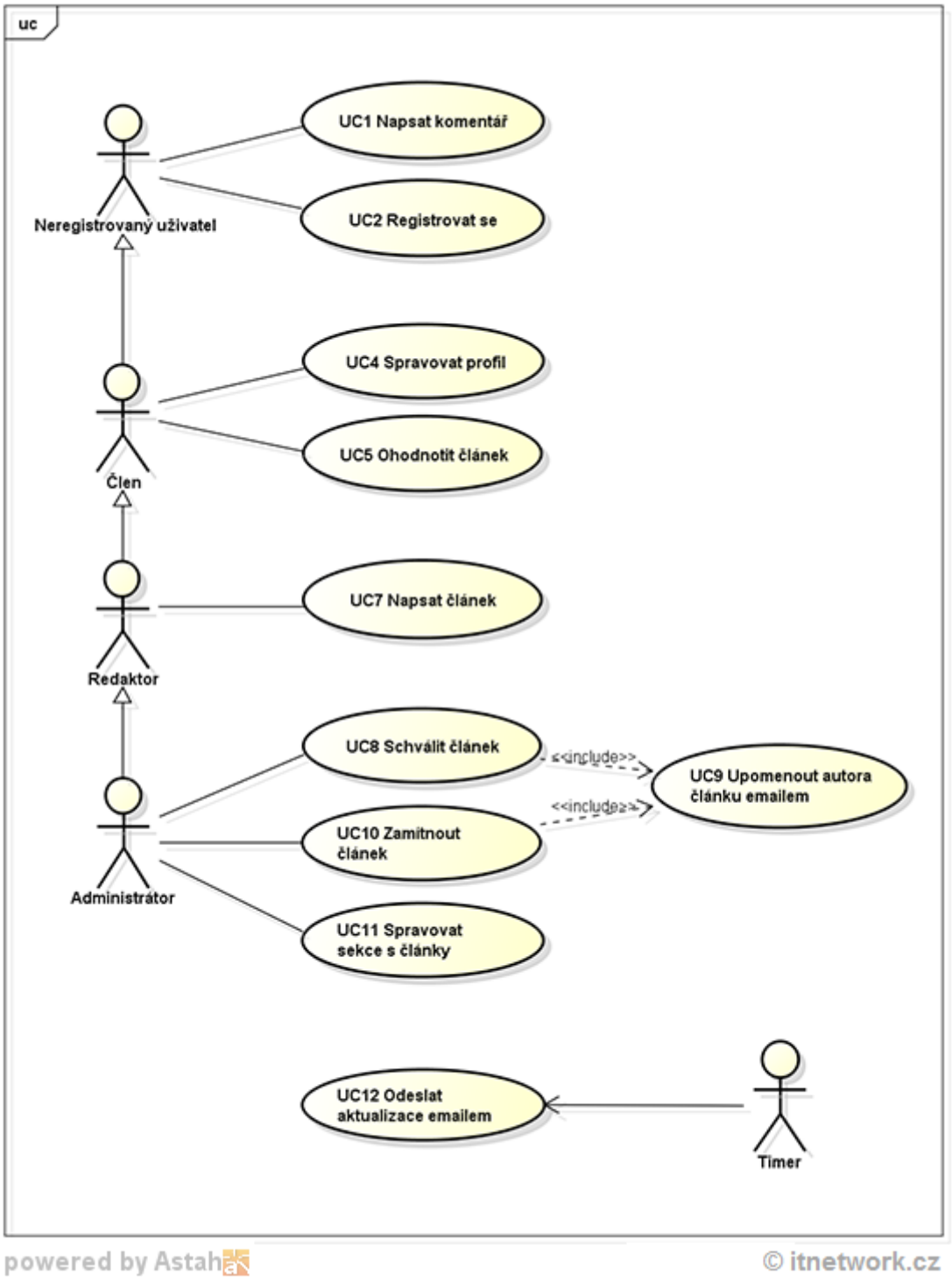


Figure 7 Source: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

7.1. Software testing and installation

There are a lot of methods of software testing, while each testing method usually focuses on a different testing area. Individual methods can be divided into functional and non-functional testing. Functional testing focuses on meeting all the user requirements. Targets of the tests are defined in ISO/IEC 12207 Standard or can be obtained from results achieved by means of FURPS+. Each requirement is supplied with a test.

On the other hand, informal test cases do not directly test the system functionality; they are based on normal expected operations, which the system should be able to ordinarily perform. These are as follows:

- Security testing - tests the system security
- Load testing - tests putting demand on a system and measures its response.
- Usability testing - measures conditions under which the system can be used. For instance, internet speed connection.
- Documentation testing - user documentation is tested in regard to its comprehensibility and consistency. Within development documentation, the amount of coverage is tested.

Software systems are also tested regarding their verification and validation.

- **Verification** answers the question: whether the developed product meets all the technical requirements.
- **Validation**, on the other hand, answers the question: whether the software corresponds with the intended use.

For example, if a customer wants E-shop system, our developed system has a lot of entertaining functions; however, there is a little opportunity to see them so that customers do not usually find them and thereby do not buy them.

Sometimes we speak about Black box and White box testing.

- **Black box** testing approaches the developed software system as a black box. Therefore, it does not deal with the inside. It examines the functionality only from the end-user's perspective. It usually checks whether given inputs correspond with the required outputs. It also tries to find unusual or unexpected inputs which might not adequately respond.
- **White box** testing, on the other hand, is familiar with the program structure. It examines whether separate program functions work properly. It tries to test each function separately.

Vondrák (2002) advises following activities to deploy software system to the user:

- Development of the final product or its versions

- Completion of the software system
- Distribution of the software system
- Installation of the software system to the user
- Assistant services to users
- Planning management of beta testing
- Migration of existing data and software products

8. Measuring qualities of software systems

If we succeed in meeting user's/customer's requirements at the end of the whole process, quality software is likely to have been developed. All the same, this indicator is highly subjective. As a matter of fact, each user/customer has different expectations. It is therefore necessary to adopt more objective criteria for assessing the quality and benefits of software systems.

These criteria are called metrics. (Metric may also refer to Metric - generalization of physical distance in mathematics)

Žáček (2017) defines the concept of metric as follows:

A metric may be defined as such: "A metric is a clearly defined financial indicator or non-financial indicator or evaluation criterion that is used for evaluating levels of efficiency within the particular area of business performance and its effective support by means of IS/ICT." The group of metrics associated for a certain purpose (i.e. Related to a specific area, process or project) are called "portfolio metrics".

Metrics are usually divided into hard and soft metrics.

- Hard metrics - are objectively measurable , for example average response time, the number of error per time unit or warranty period
- Soft metrics - are more subjective and very hard to be objectively measured. They can be evaluated on a scale typically from 0 - 100 in the order of evaluated products or alternatively a verbal evaluation can be carried out

In addition, there are plenty of systems for managing the quality of software system development. By and large, customers like when their products have the quality certificate. In fact, they like to pay extra money for that. These systems help minimize errors and boost the efficiency of the development process. Some of them will be briefly suggested in the following text.

8.1. CMMI

Capability Maturity Model (Integration). This model is remarkably widespread in the USA and Japan. It is designed for development teams. This model is relatively elaborate thanks to which it can serve as a guide for enhancing the quality of the development team. It consists of set of rules and recommendations which should be observed for a rapid development and designing of new products. Furthermore, it focuses on an organization, planning and monitoring of development processes. It is specified by dividing development teams into five maturity levels and capacities. In this way, individual teams can move within these levels. Maturity levels are assessed by a well-trained evaluator in an exact procedure.

Maturity levels are (according to Wikipedia)

- Initial: On this level, teams perform these processes either not at all, or only partially.
- Managed: Project management is defined and activities are planned.
- Defined: Procedures are defined, documented and managed.
- Quantitatively Managed: Products and processes are quantitatively managed.
- Optimizing: The team continually optimizes their activities.

Capacity levels

- Incomplete: Some activities are not performed.
- Performed: Activities are performed.
- Managed: Activities are performed according to their management.
- Defined: Activities are performed according to their definition.

8.2. ISO 9000:2001

In contrast to CMMI, it is only loosely defined. It defines quality management system. This standard allows specific organizations to demonstrate their capacities for production and distribution of products. It is thereby only loosely defined. Thus, the development process is very specific. For this reason, English-Swedish TickIT or ISO/IEC 90003 interpretation were made.

8.3. Six Sigma

It is a set of techniques for process management whose aim is to understand and continually and regularly improve its efficiency, or to meet the customer's requirements by means of understanding customer's needs, process analysis, standardization measurement methods and their analysis using statistical methods.

Basic philosophical principles of this methodology are:

- A sustained effort to achieve stable and predictable results of the process is vital for a commercial success.
- Production and business processes have qualities that can be defined, assessed, analysed, improved and controlled.
- In order to achieve a constant quality improvement, the whole organization must be engaged, including the top management.

Main techniques for example include DMAIC cycle - it is meant for understanding and

management of processes.

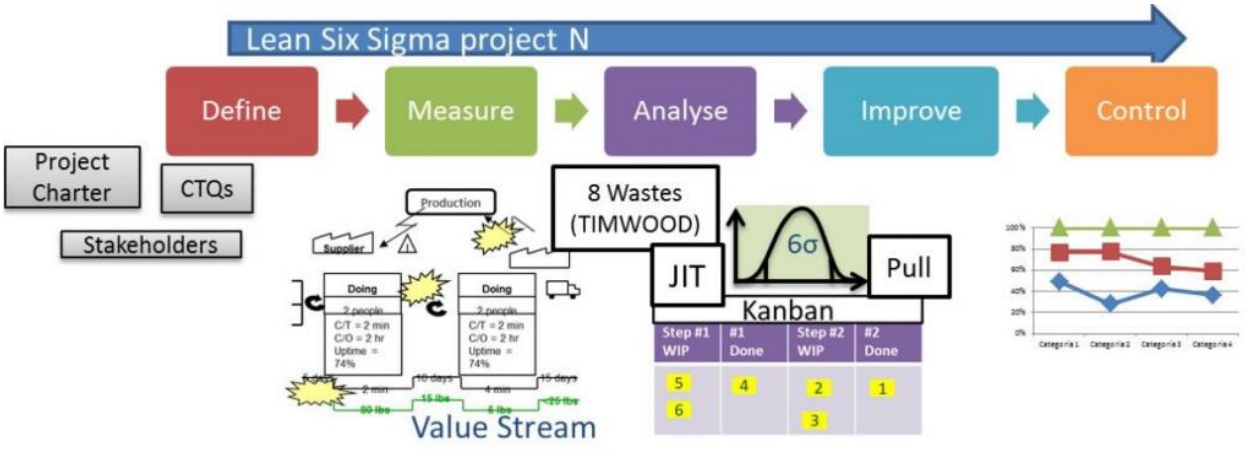


Figure 8 Source: http://www1.osu.cz/~zacek/infs2/02.pdf

DMAIC involves 5 phases.

- **Define** - defining an actual and ideal process. It suggests a way from the actual to the ideal process.
- **Measure** - measuring the current process. It means, for example, the average waiting time which provides us with "Hard Data" describing "objective reality".
- **Analyse** - the measurement results are statistically analysed in order to produce a working model of a particular project; alternatively, the analysis tries to find out which parts of the process need to be improved.
- **Improve** - the implementation of thoughts coming from the previous phase. Major goals include satisfying the customer, increasing efficiency, reducing costs.
- **Control** - if a change has been successfully introduced, it needs to be standardized and controlled regarding whether or not it brought an improvement.

This phase was originally brought in by Motorola Company. Nowadays, this phase is made use of by for example Honeywell, HP, Texas Instruments, NASA and other companies. It is based on applied statistical procedures supplied with qualitative tools.

9. Maintenance and operations of software systems

Until now, we only dealt with software development. All the same, the software development only marks the beginning of the software life cycle. It is mainly its operation that constitutes its crucially important part. To put it more precisely, if we developed a powerful software system which would after one month of its operation regularly collapse as a result of its poor maintenance, the customer would surely feel that the whole system was poorly developed. It is therefore necessary to carry out a regular maintenance for the software to operate effectively.

ITIL focuses on the techniques of maintenance and operations of information technologies. It is a set of concepts and techniques for a more effective use of IT services. IT services involve IT systems which key goal is to stimulate business processes (i.e. for the enterprises to do their work properly).

ITIL is actually a volume of book publications which contain invaluable experience in the field of Service Management of information technologies. It thereby establishes a framework for IT service management which tells us when and what to do. ITIL concept is actually very simple; or more precisely, why we should design and develop the whole process from the beginning, when a lot of other companies have it already introduced and regularly improved.

ITIL use proactive approach. It deals with problems in advance, not retroactively. If a problem has already occurred, it tries to find it out by active detection for which it establishes correct procedures. The whole process is regularly managed, monitored, assessed, evaluated and constantly improved. All these processes should provide the customer with an added value. However, the terminology can be a problem in some cases where companies use different technical terms. ITIL thereby strives for standardizing the terminology. These procedures are established as platform-neutral.

The biggest advantage of these approaches is probably their efficiency, thanks to which, for example, the duration of IT system blackouts can be reduced.

For that reason, two basic processes **Service Support** and **Service Delivery** have been introduced. Within these processes a customer contact point - Service Desk has been introduced. This point collects customers or users' requirements. It also registers IT processes, which means that it can respond and fulfil the requirements. It also provides basic IT support.

In case of a problem, **Incident management** process, which tries to minimize unpleasant consequences for customers concerning problems with IT systems, takes place. One of the most important criteria is the speedy solution of the problem.

Moreover, Incident management is supported by **Problem management**, which administers the register for dealing with problems. It at the same time analyses existing problems and their character and, if appropriate, it introduces structural changes of IT sys-

tems.

Individual changes are registered in **Change management**. Release management further plans and manages individual changes of IT services (release)

ITIL Version 3 consists of 5 parts and one introductory book; (Žáček, 2017) characterizes them as follows:

- Service Strategy - deals with harmonizing business and IT, management strategy of IT services, planning.
- Service Design - deals with IT services, process planning (creating and maintaining IT architecture and procedures).
- Service Transition - transits IT services to business environment.
- Service Operation - delivers and manages process activities, application management, changes, operation and metrics.
- Continual Service Improvement - deals with IT services, improvement competences, methods, practices and metrics.

10. Literaure

Eysenck, M. W., & Keane, M. T. (2008). Kognitivní psychologie. Praha: Academia.

Nolen-Hoeksema, S. (2012). Psychologie Atkinsonové a Hilgarda (Vyd. 3., přeprac.). Praha: Portál.

Sklenář, V. (2007). SOFTWAROVÉ INŽENÝRSTVÍ [Online]. Retrieved June 29, 2018, from https://phoenix.inf.upol.cz/esf/ucebni/syspro.pdf

Softwarové inženýrství [Online]. (2001-). In Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation. Retrieved from https://cs.wikipedia.org/wiki/Softwarov%C3%A9_in%C5%BEen%C3%BDrstv%C3%AD#Softwarov%C3%A1_krize

Plháková, A. (2004). Učebnice obecné psychologie. Praha: Academia.

Rational Unified Process [Online]. (2001-). In Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation. Retrieved from https://en.wikipedia.org/wiki/Rational_Unified_Process

Rychta, A. (2011). Softwarové inženýrství [Online]. Retrieved June 29, 2018, from http://www.ksi.mff.cuni.cz/~richta/NSWI026/NSWI026-1-Uvod.pdf

Říčan, J. (2016). Používané metakognitivní strategie žáků pátých tříd ve specifické doméně čtení (Disertační práce). Praha.

US Department of Justice (2003). INFORMATION RESOURCES MANAGEMENT Chapter 1. Introduction.

Vondrák, I. (2002). Úvod do softwarového inženýrství [Online]. Retrieved June 29, 2018, from http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf

WHITE, Stephen A. Business Process Modeling Notation [online]. [cit. 2018-06-26]. Dostupné z: https://is.muni.cz/el/1433/jaro2014/PV165/um/46771256/pr_06_bpmn.pdf

Žáček, J. (2017). SOFTWAROVÉ INŽENÝRSTVÍ [Online]. Retrieved June 29, 2018, from http://www1.osu.cz/~zacek/sweng/skripta_sweng.pdf

WEB APPLICATION BASICS

1. Communication, Networks, Protocols

1.1. Communication protocols

In the context of computer networks, we often see communication protocols. They precisely define the way of communication to perform a specific function across all levels. There are protocols for sending data, creating secure channels, and searching for IP addresses corresponding to the domain name, or delivering emails, etc. The protocol is known to both communicating parties and precisely describes what content, what sequence and what timing is used for transmission. Any deviation from this structured communication may be interpreted as an error.

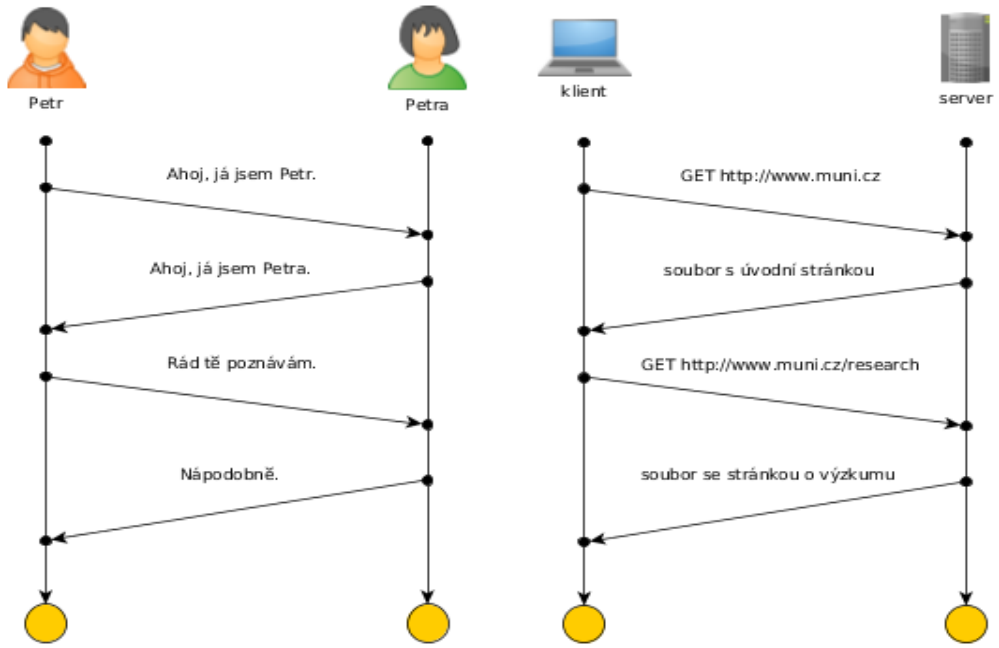


Fig. 1: Communication protocol illustration

1.2. TCP/IP

The essential principle of the computer network architecture is to divide communications into layers by abstraction. Each layer is responsible to describe the transmission starting the application level through physical data link communications. The TCP/IP network model is the cornerstone of all present networks as well as the entire Internet. The name is derived from two major protocols ensuring data routing and transport between nodes. The IP protocol describes the addressing of nodes, data broken down into packets and their routing inside the network.

The communication between the same layers of two different systems is controlled by the communication protocol using a data link created by the adjacent lower layer. The architecture allows to exchange protocols of one layer without impact on the others. The TCP/IP architecture is broken down into four layers (unlike the OSI reference model with seven layers):

- Application layer
- Transport layer
- Internet layer
- Network interface

Fig. 2: TCP/IP network model layers

TCP is a transport protocol building a "linked" service on IP transmission: it ensures the control of successful transmission and potential forwarding of missing or damaged parts and selects an optimum transmission speed.

The TCP/IP protocol family which is currently used for the absolute majority of network communications was designed between 70s and 80s with the aim to create a more robust model of network communication, to a certain extent able to cope with losses of parts of the network. The key (and quite revolutionary at that time) idea is *packet switching*: data are not transmitted as a continuous stream, but in separate blocks (packets) and each network node alone determines which way to send/forward packets.

In practice, this idea is based on a set of protocols implementing the necessary functions. Protocols are typically broken down into several levels (layers). Instead of the abstract ISO/OSI model working with seven layers, a simplified five-layer model is mostly used for TCP/IP interpretation (some protocols in the TCP/IP model implement the function of multiple layers of the ISO/OSI model).

1.2.1. APPLICATION LAYER

The application layer is the upper layer which refers to application protocol data of the individual network services. It includes network application protocols: electronic mail, HTTP (websites), DNS (domain service) and others. There is a huge number of such protocols, for example HTTP, SMTP, FTP, NTP are the most commonly known. For TCP/IP those are data which must be transmitted to a target recipient. The implementation is up to lower layers.

1.2.2. PHYSICAL LAYER

The physical layer is the lowest layer in the model. Unlike the upper layers it is not a software layer (protocol) and refers to a specific physical medium used for data transmission. An example can be a twisted pair, cabling in the majority of local Ethernet networks, coaxial cable, optical fibre, or a telephone line. The medium does not need to be material – e.g. in case of wireless networks in the microwave band (Wi-Fi, breezenet) or free-space optical links (FSO).

1.2.3. DATA LINK LAYER

The data link layer is the lowest of software layers. It is the lowest communication protocol used for data transmission on a physical medium. This protocol is most often closely connected with a specific medium, but this correspondence does not need to be 1:1, e.g. Ethernet is not only implemented on twisted-pair cabling, but you can see implementations using coaxial cabling or optical fibres. Another example of a data link layer protocol is PPP which is used for implementation of dial-up connection or computer serial links. The important feature of data link layer protocols is that they only deal with communication between nodes that are *directly* linked (hence the name).

1.2.4. INTERNET LAYER

The *Internet layer* is responsible for global addressing and routing, in practice most often implemented by an IP protocol (in two versions IPv4 and IPv6). Although there exist addresses in link layer protocols and in case of Ethernet MAC addresses, they are (or, at least should be) globally unique, they cannot be used for packet routing because they do

not show the target. But IP protocol addresses (IP addresses) are hierarchical so that delegation of individual ranges reflects the network topology. Therefore, you can tell from the target IP address where to further send a packet, that is at least the following hop on the route. In addition to this basic function, IP protocol allows, for example, fragmentation (chopping too big packets into multiple smaller datagrams) or packet designation by type of service (ToS).

1.2.5. TRANSPORT LAYER

UDP and TCP are the most frequently used protocols of the transport layer. UDP (User Datagram Protocol) is a minimal message-oriented transport layer protocol, only introducing the concept of port as a specific process address (better to say a socket) within the target node. But UDP is still a stateless protocol (not a connection in the real sense of the word), not dealing with the issue of lost packets or their sequence. Still it is often used because it is simple and low overhead, in particular where those issues do not need to be addressed.

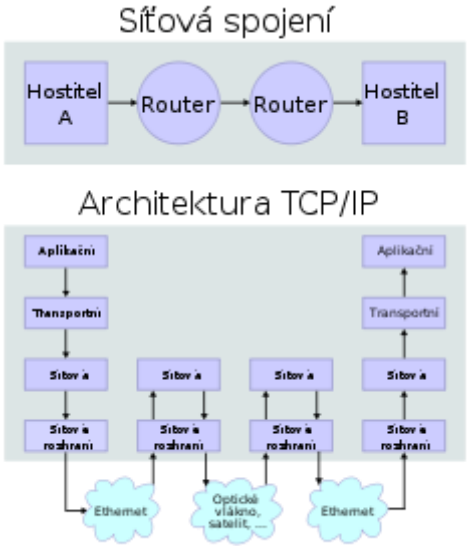
The opposite approach is represented by TCP (Transmission Control Protocol). Unlike UDP it establishes connections between two ports on the end nodes. In the view of client applications, the behaviour of this connection is similar to a communication pipe between two processes (but unlike the pipe TCP is a two-way connection). It is guaranteed that a stream (byte sequence) dispatched from one side is received in the same format on the other side. TCP protocol provides for detection and resending of lost data as well as data rearrangement from packets arriving in the wrong sequence. Since TCP transport protocols ensure fairly high levels of comfort for the application layer, they are currently used for the majority of communication. The disadvantage is a higher overhead and relatively low response to outages, therefore UDP is preferred for some specific purposes (e.g. most DNS queries or VoIP).

1.3. ICMP

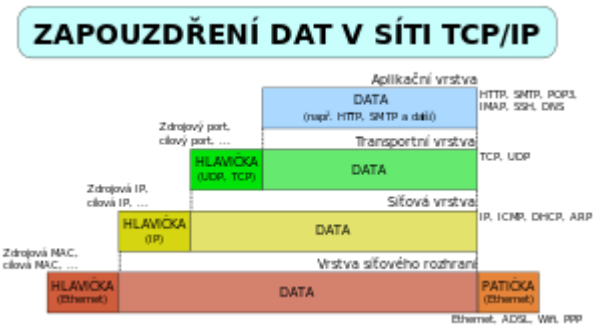
ICMP (Internet Control Message Protocol) goes a little beyond the layer structure. Packets (message) of this protocol are transferred directly through an IP protocol. However, ICMP is not considered as a transport protocol because it is not used for transmission of application data. This protocol is used for diagnostics and service purposes. The examples of ICMP applications are packet destination unreachable messages, ICMP echo and echo reply packets, or some service types of messages (redirect).

1.3.1. PACKET SIZE

The maximum (theoretical) size of IP packet is 65535 B, but the data link layer is often a limiting factor. Since most of packets go through the Ethernet (or, its equivalent) at least once, the size of packets is often selected according to its limit (1536 B), hence the most typical value is 1500 B. But of course, this is the maximum value and packets are very often much smaller, in particular for interactive applications. The size of IP and TCP headers is 20-60 B (typically near the lower limit), UDP and ICMP headers are 8 B, and Ethernet header is 14 B (in addition 2 B at the end of packet is the checksum).



The TCP/IP layers providing data transfer between two hosts through two routers.



Overview of application data encapsulation on TCP/IP layers.

In the view of problem complexity, the network communications are divided into so-called layers representing the hierarchy of activities. The exchange of information between layers is precisely defined. Every layer uses services of lower layers and provides services to upper layers.

Internet Protocol is the basic protocol of the Internet layer and the entire Internet. It generates datagrams based on network IP addresses contained in their headers. It provides the connectionless network service to upper layers.

Currently, the most widely used is the IP protocol version 4. The new version 6 which addresses the deficiency of addresses in IPv4 and security problems and enhances other properties of the IP protocol is only used by several percent of devices connected to the Internet worldwide, but the number is rapidly rising.

1.4. IPv4

Internet protocol version 4

- 32 bit addresses
- App. 4 billion of IP addresses, insufficient now
- Format: xxx.xxx.xxx.xxx where xxx is an arbitrary number 0 - 255 (8 bits)

1.5. IPv6

Internet protocol version 6

- 128 bit addresses
- Security support
- Mobile device support
- QoS - Quality of Service function
- Packet fragmentation - distribution
- Not compatible with IPv4

The Address Resolution Protocol is used to search for a MAC physical address by a

known IP address. If needed, the protocol will send a datagram with the searched IP address information and address it to all stations in the network. The node with the searched address will respond and fill in its MAC address. If the searched node is not in the same segment, the relevant router will respond and return its address.

1.6. ICMP

The Internet Control Message Protocol is used for transmission of **control messages** related to error statuses and special conditions of transmission. It is used, for example, for computer availability testing in the *ping* program, or for packet route tracing to another node by the *traceroute* program.

1.7. TCP

The Transmission Control Protocol creates a virtual circuit between end applications, i.e. **reliable data transmission**. The protocol features are:

- Reliable transport service delivering all data to the addressee without any loss and in the correct sequence.
- Connection service with phases of connection establishment, data transmission and connection termination.
- Transparent transmission of arbitrary data.
- Fully duplex connection, parallel two-way data transmission.
- Distinguishing between applications using ports.

1.8. UDP

The User Datagram Protocol provides a non-reliable transport service for applications which do not need reliability of TCP protocol. It lacks the connection establishment and termination phase and already the first UDP segment contains application data.

1.9. SCTP

A reliable protocol for transmission of datagrams in multiple streams. It is primarily used for the purpose of telecommunications. It has some additional features that TCP does not provide:

- Multihoming – the communicating node can have several IP addresses.
- Breaking down the data stream into datagrams.
- Using multiple data streams – to reduce blocking communication due to a miss-

ing data block which may happen in TCP.

- Route selection and tracing – Using an alternative if the primary address has availability problems.

2. Languages for presenting web content

2.1. Introduction to HTML

HTML is a simple markup language for creation of websites, which are just common text files containing a text and several html tags that determine the meaning and appearance of the individual website parts.

Besides html, there are several other languages used for creation of www pages: css, php, javascript, etc.

HTML is a basis in case you want to learn something about css styles (css styles determine the appearance of web pages) and later about php (related to programming).

2.2. History of HTML

Web appeared in 1989; since then it has been continuously developer. Currently, HTML 5 is being developer.

2.3. Creation of html pages

We can use a simple text editor, notepad or special editors with html code support. These programmes highlight syntax and allow us to switch between a code and preview. Working with such an editor is much more efficient than working with a common notepad. .

2.4. Terms used in HTML.

Tag – is a basic html mark, written as <tag>.

Attribute – is written within a tag and sets its property. It is written as: <tag attribute="value">.

Element – Writing of heading: <h1>Page heading</h1>.

First page.

HTML pages are just common text files with tags

Tags

Tags are written between brackets < >, some of them are paired, some are not.

Writing a non-paired tag:

```
<tag>
```

Writing a paired tag:

```
<tag>some text</tag>
```

In the case of a paired tag, it is important to write a slash, otherwise the explorer would not understand.

Attributes

Attributes are written straight into the tag.

```
<tag                                                    attribute="value">
<tag attribute="value">Paired tag with attribute.</tag>
```

It is prohibited to cross tags

Tags can be within other tags, however they cannot cross each other partly.

```
<b><i>bold italic</b></i>
```

Correct writing:

```
<b><i>bold italic</i></b>
```

Size of characters

In html the size of characters does not matter, it is therefore possible to write both **<TAG>** or **<tag>** , whereas in xhtml, which is the latest version of html, neither tags nor attributes can be written in capital characters.

In **url** it is necessary to respect the size of characters, that is, FILE.html is not the same as file.html.

2.5. HTML

HTML stands for Hypertext Markup Language. Hypertext markup language was developer from SGML and became the most widely-used language for creating web pages. In the past, the versions most often used were HTML 2.0, HTML 3.2, HTML 4.01 and HTML 5. From HTML was developer also XHTML (extended hypertext markup language) as an

application of XML, which is not of higher significance, in my opinion. In 2010, HTML 5 started to be considered; most of its innovations have already been used.

Structure of a html file

The most frequently used "template" of a page:

```
<!DOCTYPE HTML>

<html>

   <head>

                <meta charset="windows-1250">

            <title>Name</title>

   </head>

   <body>


page text


   </body>

</html>
```

2.6. CSS

History of CSS

CSS appeared around 1997. It is a collection of methods for graphical design of web pages. The abbreviation stands for Cascading Style Sheets. They are called cascading, since definitions of a style can be layered, but only the last one is valid. However, this is not important now.

There has also been designed CSS 2, improved and more sophisticated forms of styles; however, they do not work well in the most widely used Internet Explorer.

When to use CSS

In 2015 it could be said that the whole web is formate using CSS. From the previously used HTML, only boldness and italics were still used. Therefore i tis good to have some knowledge of CSS if you want to create web pages. First of all it is necessary to know how HTML works. Without any basic knowledge, it is not recommendable to start working with CSS. CSS shall be studied if:

- you want to have formated websites - colours, justify the text, columns layout,

etc.

- you often write texts for the Internet without losing time formating,
- you deal with script, especially Javascriptem
- administrate (or plan) web with many pages that are supposed to have a similar design,

2.7. Other applications of CSS

There are three possibilities of using CSS

The style can be decrared in three ways (see the examples below). It is enough to know at least one of the following methods:

In the source text for a formated element by means of the attribute style="...". This is referred to as **direct style**. It is quite unsuitable.

By means of "**style sheet**" in the heading of the page. Style sheet is a list of styles, giving information about formating (e.g. green headlines). In a page, a style sheet is written between tags <style> and </style>.

By means of external style sheet -- it is a **file *.css**, that the page is linked to with a tag <link>. The file contains a style sheet. The main advantage is that many pages can be linked to one file and all the pages have similar design.

Examples

Make a paragraf in red font using CSS. Three possible ways can be used:

Direct style

In the source, this paragraph declaration is written:

```
<p style="color: red">This paragraph will be red.</p>
```

Explanation: <p> is a mark designating a paragraph. Attribute "style" is a general attribute that could be used for each element.

Style sheet

In document heading, a style sheet is written between tags <style></style>:

```
<style>
  p {color: red}
</style>
```

Paragraphs are written in the body of the page:

```
<p>This paragraph will be red. </p>
<p>This paragraph will also be red, because all paragraphs will be red.</p>
```

If we want only some paragraphs to be red, use "classes" and "identifiers".

By external CSS file

There will be a file created, named e.g. styly.css. It will contain only the following text:

```
p {color: red}
```

In the heading of a html document we want to change by style, the following link must be written:

```
<link rel="stylesheet" type="text/css" href="styly.css">
```

In the body of the document, all paragraphs will be red.

2.8. Syntaxe

As you may have noticed, CSS are not a part of HTML; therefore they are written in a different way. If the table below appears to be too theoretical, please pay attention only to the examples in the lower part.

Direct style:	`<tag style="characteristics">styled element</tag>`

Style sheet:	`<style>`
	`tag {characteristics`
	`2nd tag {characteristics}`
	`</style>`

Characteristics simplified:	`characteristics: value; 2nd characteristics: 2nd value`
characteristics in general:	`characteristics: value [, value2] [; another characteristics]`

Examples:

Direct style	`<p style="color: red;">text of red paragraph</p>`

Style sheet	`<style>`
	`p {color: red}`
	`body {background-color: yellow;}`
	`</style>`

Simple writing of characteristics	`color: red`
complex writing of characteristics	`font-family: Arial, Arial CE, sans-serif;` `color: red;`

It shall be note where quotation marks, colons, curly brackets, semicolons, and commas are used. Example of the a correct writing:

h2 {color: green; background-color: yellow}

spaces and line endings do not play a significant, they can be added and left out. The characters size does not play a role. There is a list of characteristics and their values available.

The explorer ignores the values it does not recognise.

In style sheets, comments are made similarly as in Java between /* a */. Two slashes do not work.

Example with a headline

It is quite easy in style sheet or external css file.

```
<style>
    h1 {color: green;}
    h2 {color: blue;}
</style>
```

This way, the entire document will contain green first level headlines and blue second level headlines, however, only supposed that the tags <h1> and <h2> were used for writing a text. In other words, style sheets can be used only in well-structured texts.

2.9. CSS styles

CSS styles are cascading, they are used for creating a style of a web page (colour, font, font size). Using CSS, one file can influence the design of the whole web.

Outdated methods

Before CSS styles, for a web page style, the element<font face="font" color="color" size="size">, was used, that is no longer used. Compared to CSS, it has the following disadvantages:

- If you often changed the style of a text, this tag appeared in the source code very often, which slowed down the page.
- It enables to change font, colour, and size only <font face="font" color="color" size="size">

2.10. DHTML

Maybe you have already come across the abbreviation DHTML, or Dynamic HTML. This language is created nearly exclusively from JavaScript, VBScript (language with characteristics similar to JavaScript) and CSS styles. This language uses the advantages of HTML, JavaScript, and CSS, thus creating a perfect design and good-looking pages.

2.11. CSS styles and classes, identifiers and style

CSS - Cascading Style Sheets – cascading styles were first implemented by Microsoft in 1996 into Internet Explorer 3.0. CSS style completely removes <font> and introduces <style>. By means of CSS styles, it is possible to define colour, font, size, and many others (box, underlining, boldness, waviness, display, bullets, margins..)

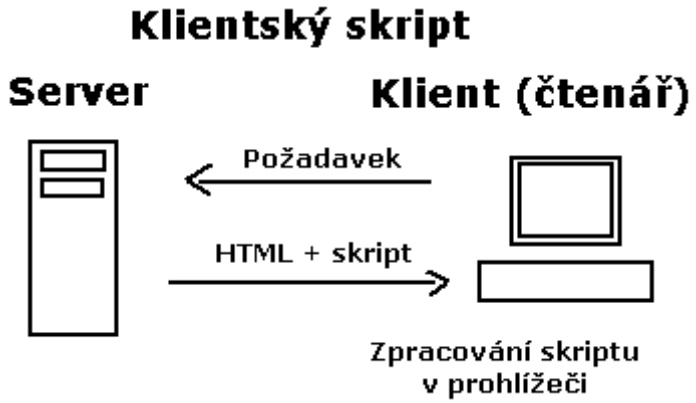
CSS styles are applied mainly by means of classes and identifiers. These enable to create a CSS style by means of one attribute only and the user do not have to repeat one code ten times. In addition, it is also possible to define the style of the elements (h l, table, etc.) by means of selectors. For example, each element <input> will always have a red text – this is possible to make with one CSS rows.

3. Client-side logic - JavaScript

3.1. What is JavaScript

JavaScript is a programming language used on websites. It is written directly into HTML code, which is a great advantage due to its simplicity.

JavaScript is a client script. This means that the programme is sent with a page to the client (into the explorer) and it is performed there (unlike server scripts, which are performed on a server and the client gets only the results).



Legend: klientský skript – client script, klient (čtenář) – client (reader), požadavek – requirement, zpracování skriptu v prohlížeči – processing script in explorer

There are also other languages of client scripts, e.g. VBScript. However, there are so rarely used, that when you mention “scripts” it mostly means “JavaScripts“.

JavaScript is not Java

JavaScript is often confused with Java. Java is a separate programming language. It only has a syntax similar to JavaScript.

Necessary skills

- HTML, basics of HTML at least
- Basics of programming

3.1.1. THE LANGUAGE CHARACTERISTICS

JavaScript is a language

- interpreted – it does not have to be compiled
- objected – it uses the object of an explorer and built-in objects
- dependent on explorer – but works in most explorers
- case-sensitive – the size of characters in the ZÁPIS matters

- its syntax is similar to C, Java, and the like

Limitations of the language

- JavaScript works only within an explorer.
- Users can block JavaScript
- There are various different versions of the language and explorers, which causes frequent errors.
- it cannot access files (except for cookies) or any other system objects.
- It cannot save data (except cookies).

This makes it only a secondary language, applicable only for HTML pages.

How to deal with JavaScript

After mastering the basics, it is recommended to notice scripts on other websites. Most scripts are written directly in the source code of the webpages, so it is possible to copy it (some codes are in external files, but even those can be downloaded.

3.1.2. EXPLANATION OF SCRIPT

Script is written into HTML between the tags <script> and </script>. Everything written between the tags is a programme written in the language Javascript.

The example uses the command **document.write()**. This enables normal writing into the document stream. The written text is immediately displayed in the explorer.

Normal text must be written between quotation marks (unlike variable). A line must not be wrapped between the quotation marks.

Each JavaScript command is ended with a semicolon or wrapping a line.

How to create the first script

Everything created within JavaScript is called script. It can be placed in the page or it is possible to create a link to it. In such a case the page uploads in the JavaScript page. Separate files written in JavaScript have the extensions .js or .jse. The extension .js is more common. The only thing you need for creating a script is a source text editor (PSPad, text editor or any HTML editor). For browsing you need an explorer (at least Internet Explorer and Mozilla Firefox so that you can check scripts in these most commonly used explorers).

Inserting a script into a page

Script is written between the tags <script> and </script>. They can be inserted between the section "body" or "head" (it depends on the purpose of a script).

```
<html>
        <head>
            ...
            <script type="text/javascript">
            .. javascript script body ..
            </script>
            ...
        </head>
        <body>

document body..
            <script type="text/javascript">
            .. javascript script body..
            </script>

document body..
        </body>
</html>
```

Tag <script>

Syntax of the tag <script> is the following:

```
<script type="text/javascript" src="url of external file">
   <!--
                javascript script content
   //-->
<script>
```

Attribute type designates a type of a script (in the case of JavaScript, "text/javascript"). Since there are explorers that do not understand JavaScript, it is recommended to write <!-- at the beginning of the script and //-->, at the end of the script, otherwise the explorer would write the script as a normal text (now, it will consider the text a comment and will not display it).

3.1.3. SCRIPT WRITING

In JavaScript it is important to distinguish between capital and normal letters (it is case-sensitive), therefore document.write is not the same as DOCUMENT.write. This rule must be observed, otherwise the script will not work.

JavaScript is a multi-platform, object-oriented scripting language, whose author is Brendan Eich from the company Netscape.

Now, JavaScript is usually used as interpreted programming language for WWW pages, often inserted directly into a HTML page code. It usually controls various interactive elements GUI or creates animations and image effects.

JavaScript was originally a trade name of an implementation of Netscape which was originally developed under the name Mocha, later LiveScript. It was announced with the company Sun Microsystems in December 1995 as a complement to the languages HTML and Java. For the version of the company Microsoft, the name JScript is used. This is supported by the platform .NET.

A programme in JavaScript is usually started after downloading a WWW page from the Internet (on client side), unlike other interpreted programming languages (e.g. PHP and ASP), which are started on the side of server even before their download from the Internet. This implies certain safety limitations. For example, JavaScript cannot work with files, otherwise it could threaten the privacy of the user.

JavaScript can be also used on the side of a server. The first server-side implementation

of JavaScript was LiveWire of the company Netscape launched in 1996. Nowadays there are more possibilities, including open-source implementation Rhinola based on Rhino, gcj, Node.js and Apache.

Besides DHTML, JavaScript is used for writing of extension for many applications, e.g. Adobe Acrobat.

JavaScript can be also used in the operating systems Windows using the programme Windows Script Host and thus replace batch files MS-DOS.

4. Web architecture

Web architecture design

We never start to create web with graphical design. Firstly, the objectives of the individual pages are defined and their location on the web is specified.

Why is it important?

Thanks to the web architecture design, it is possible to imagine what the resulting web will look like and what his functions will be.

This way it is possible to identify and correct potential defects, thus saving money it would require to repair programmed applications later.

Web structure design

In the structure design we have to define all web pages, their mutual links and usual user scenarios. It is necessary to make sure that the information is structured appropriately, easy to use and leads to objectives achievements.

Web navigation design

A suitable navigation is an important part of a web. It enables a visitor to find quickly what they are looking for. This shall be paid great attention to when designing a web.

Wireframe modelling

Once we know the content of the web, wireframe modelling can be started, which is the basis for the graphic design of the web.

4.1. Web applications programming

Web application usually means a server-side script (a code ensuring the programme function). It is often connected with a database, a system storing web applications data (in a simplified way, a database can be imagined as an MS Excel file). The script output is a web page, which is passed to the explorer for display.

Schéma webové aplikace

Legend: schéma webové aplikace - web application graph, skript - script, www stránka - www page, databáze - database

The task of web applications is mostly to increase the interaction of the Internet presentation with its visitors or to facilitate the web administration, i.e. to spare repetitive work in creation of www pages.

Depending on the functionality requirements, a web application can consist of just a few lines of code (e.g. when sending contact forms), but there are also web applications consisting of thousands of lines.

More complicated web applications are often linked to other software within a company, e.g. to ordering systems, accounting programmes, etc. This enables to save costs of hu-

man work. It is also possible to connect the application to online payment systems.

Examples of simple web applications

- Contact form
- Guestbook
- Discussions or chat
- Catalogues and price lists
- Dictionaries
- Banner systems

For larger presentations it is better to tackle the whole presentation in a dynamic way, e.g. using templates or deploying the content management system. A specific form of such an application is internet shop. Due to the many advantages they offer, weblogs are becoming increasingly popular, as well as various intranets and extranets. All of them belong to special types of web applications.

When programming all web applications, the following aspects shall be taken into account:

- **Safety** – is a priority for any web application, as there is always a risk of losing or destroying of data. There is also a risk that the data or information will be stolen by an unauthorised person or that the web server will be hacked by means of an application (a risk of losing company image).
- **Using available resources** – in programming each web application we try to use finished pieces of codes from other resources; for this purpose we own an extensive scripts archive. This way it is possible to spare some work necessary for the development of the application and thus also money and the total time required for the realization of the order.
- **Scalability** – when the web application is proved to work well in practice, usually it starts to be modified, improved and extended. If these modifications are considered in the application design, their incorporation is much easier and cheaper. For this reason, most complex web applications are tackled in a modular way.
- **Speed** – slow web applications is not very applicable, it is also a problem for explorers. Therefore, we try to optimize all scripts to be fast, and therefore, it is also recommended to deploy a finished application to our servers. Since there are al-

so the latest development tools installed, which are missing in many servers intended for commercial webhosting, the development of the web application is also faster and less expensive.

- **Maximum load** of web application is a term associated with high traffic. If a web is visited by a higher number of visitors (e.g. when you expect a product to be presented), the server is often incapable of serving them (we say that the server has "crashed"). The capability of a web application to withstand the high load requires choosing suitable tools, database optimisation, calculations and other special techniques, such as *pre-caching*. Based on the experience, we can say that the low load resistance is evidence of low capabilities of the programmes who created the web application.
- **Testing** – before starting a web application, all its functions shall be tested on a development server. This has the same configuration as the actual server, which enables a reduction of possible problems during the actual start-up.

4.2. Technology of our web applications

Currently, the world´s most widely used scripting language for web applications programming is PHP and database MySQL. This combination with a web server (programme) Apache is called **triad.** It has been proved to be useful for its flexibility. Other advantages of this system include the accessibility of functions and fragments of codes, as well as the continuous development of these programmes.

If there are reasons for it in terms of web applications or simplification of their development, other programming languages are used, e.g. Perl, Python or database PostgreSQL. Each of them is suitable for the specific needs of a specific web application.

The greatest advantage of all mentioned technologies is their inclusion in open source. It means that they are **free** (which is one of the significant factors of their popularity and widespread).

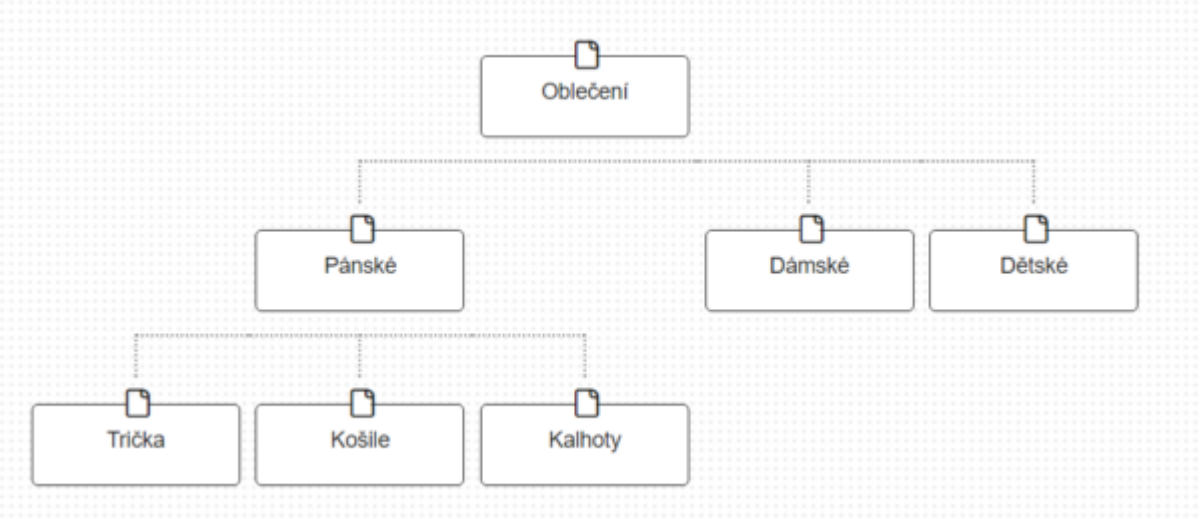
4.3. What is web architecture

Information (content) web architecture is a method of arranging the web information into a logical whole.

It is clear that the information will not be on the same page, but the web will contain more pages arranged in various levels, starting from the general to the details.Everything is thematically linked.

Example: a web selling clothes is arranged as follows: men´s clothing > shirts > short sleeve. The way the information is arranged within a web is called web architecture.

For more information, see the figure below:



Legend: oblečení - clothing, pánské - men´s, dámské - women´s, dětské - children´s, trička - T-shirts, košile -shirts, kalhoty - trousers

Why to deal with web architecture

Good web architecture is important for the following reasons:

- It enables a user to achieve its goal (the desired web content).
- It creates a logic of the whole web, perceived and used by both users and explorers and search engines
- It enables to arrange the web content so that it is easy to find in search engines.

When preparing a new web or e-shop, the content (information) architecture of a web shall be dealt within the basic web design (before graphic design and realization of the web - see 4 steps of a professional web design).

Good web architecture enables to arrange properly all the important topics. This is reflected in easier orientation as well as in better traceability by specific key works in search engines. As it is already known, traffic of search engines is free and is highly relevant.

4.4. How to create a perfect architecture

Within MD webdesign, web architecture is created based on the keywords analysis. This means to find out what keywords the users look for in Google and Seznam search engines, to analyse them and then to categorize them to get better insight.

If we want to be traceable by a specific keyword, it is recommended to have a web page on the topic of the keyword and the keyword itself.

Keywords analysis enables to find out what the search engines users are interested in.

Including these topics (keywords) in web architecture significantly increases the chances to get better positions in searching.

When creating information architecture, the content of the individual pages is also described. Web content is designed on the basis of analysing customers / clients in the form of so-called person (for more information about persons, see Wikipedia). Thanks to this, it is possible to design a web that would meet the expectations of concrete customers. And we are able to provide information they need.

Creation of web pages: starting with information architecture

Information architecture is a plan for sorting information on the web. For good functioning of web pages, great attention has to be paid to creation of a clear information architecture.

As an architect working on a house design works with a space, light, and shapes, information architect works with information, structure, and priorities.

Generally, a well design information architecture shall be understandable even in the text form. Using colours or arrows does not help if the information web structure is not understandable.

The main reasons for confusing web include: the terminology used, arrangement requiring professional knowledge, illogical arrangement of information.

In order to avoid the problems mentioned above, information architecture design shall include the following steps:

- Identification of target group
- Gathering information
- Grouping of information
- Prioritization
- Creation of information architecture

The individual steps:

1. Identification of a target group

Target group shall be identified before the information architecture is designed, as it changes dramatically the view to the information published.

In the case of public administration sites, the target user is almost everyone; therefore it will be dificult to define the knowledge or habits that most users will share. However, we are able to suppose which knowledge the users will NOT have.

Practice shows that the users accessing the websites from towns and municipalities usually do not know the difference between the terms "town" and "authority". They do not know the terminology and for them it is difficult to understand the organisational structure of an authority. The situation is even worse due to the fact that the same agenda is usually managed by different departments in different towns.

It follows that the internal structure of an authority is not a good example for creation of informational architecture presented to the general public.

2. Gathering information

The second step in designing information structure is to specify which information the visitor will look for on the web. Public authorities are obliged to publish certain information but it is not all information the web shall contain.

Public administration webs usually contain a huge amount of information, ranging from minutes of meetings, decrees, budgets, decisions, regulations to current events in the municipality and list of cultural or sports events. .

A part of this step is also a decision on which information will NOT be published on the web. It is always necessary to consider which information the web operator will be able to update regularly. The problems with outdated information can be avoided when designing the information architecture.

3. Grouping of information

If we know which information the web shall present, thematic units shall be created. These units shall contain all the information gathered in the previous step, without overlaps if possible.

At this moment at least one representative of general public shall be involved in the process. As an office worker you are aware of the structure of the office and the managing process. For public, this is different; it is not practical to expect a visitor to be aware of the inner structure of the office on a web.

The visitor goes through the web "from the top": they start from a general level and gradually specify their query with further clicks until they find the desired information. However, when designing information architecture, the process is "from the bottom" - information is sorted and categorized into groups and units. This discrepancy can be misleading for a visitor, as they do not know the web content and do not know whether the information IS on the web. Therefore, it is essential to test the understandability of the information structure "from the top".

4. Prioritization

So far, common sense and testing with users was enough. Prioritizing in the information architecture design is, however, connected with analysing. Therefore, it is necessary to have concrete numbers at disposal, e.g. the analysis of the most searched terms on the web. Priorities can also change over time, e.g. in the summer, there are frequent queries about swimming pools etc., while in the winter, the users are interested e.g. in snow conditions.

There are many methods of prioritization: e.g. placing the information at the top of the structure, placing it on the front page or main page of the thematic unit, highlighting by means of contrast, colours, size, etc.

For example, pages focusing on municipal waste shall contain information about the current price and due date of the charge without forcing the visitor to read the pdf. files

with a notice. As it is frequently sought information, it can also be highlighted with a larger font.

5. Creating information architecture

When creating information architecture, a suitable tool can be used, but a tool itself is not enough. Information architecture resembles a tree.

5. PHP /basics/

PHP is a programming language working on the side of server. PHP enables to store and change website data. The original meaning was Personal Home Page. It was established in 1996 and it has undergone many changes. Nowadays, the abbreviation stands for Hypertest Preprocessor.

PHP language

PHP is one of the most widely used programming languages used for creation of web applications. PHP is a server-side and servers for generating a HTML/XHTML page code, which is later sent to explorer (unlike client-side JavaScript, which works only after its displaying in the explorer).

The main advantage of PHP is its independence on the platform (Windows, Linux, Unix...). Other advantages include a wide range of applications. PHP can work with files and many different databases, it can generate and edit graphics, send and receive mails, create PDF, support all important internet protocols...

Since PHP has relatively free syntax (the way it is written), it is easy to learn, especially if you have experience with other programming languages. Together with web server Apache and database MySQL, it creates so-called triad, three programmes most widely used for generating web pages. This brings another advantage of PHP - a huge number of fragments, user-defined functions and ready-made solutions to common problems on the Internet.

Possibilities of PHP

PHP is not difficult to understand, and knowing the basics is enough. It can save, change and delete data. Everything is done within a web server (where there are source codes of web pages). PHP script is first executed on the server and it sends only the result into the explorer (it means that it first calculates 300/30 and then it sends number 10 into the explorer). Therefore, in the source code, there is only "10" (unlike JavaScript, which calculates directly in the explorer). Unlike JavaScript and HTML, the PHP source code is not displayed in the explorer.

PHP can be used for creating a discussion forum, guestbook, counter, opinion poll, graph; using a simple code, it is possible to dispose of the entire content of the web. Moreover, it is possible to connect the pages with databases, e.g. MySQL.

Purpose of PHP

There is at least one function that every web could use. On web pages, certain parts are often repeated: header with links, menu, footer. With PHP, it is easy to create a template for a web where the files with menu and footers can be inserted. This way it is possible to write menu only once and copy it into other pages. And changing the menu is very easy (see PHP menu).

PHP files

Web page with PHP elements has mostly the extension .php, but there can be also other extensions, e.g. .phtml, php3, php4, php5. Some hostings determined which version of PHP script to start based on the extension (current version is 7). However, this is an exceptional case; most often, .php is enough.

Installation

PHP is a language that does not work only with a certain version of an explorer (unlike HTML or JavaScript). It must be installed in the computer. The basis is a web server and libraries. To support PHP, it is necessary to install it and configure a server (usually Apache). It is recommendable to use PHP Tread for installation of PHP programme.

Webhosting with PHP

Not every webhosting includes PHP support. Support for webhosting is an above-standard service for extra charge. However, it is possible to get free webhosting with PHP support (e.g. Webzdarma.cz, PHP 5). When choosing webhosting for PHP pages it is necessary to read carefully what the offer contains.

PHP – basic information

Dynamic pages (i.e. pages first generated by a server and then sent to the client) are nowadays an essential part of any more complex website. The main scripting languages used for creation of such pages include ASP (Active Server Pages) and PHP. In the following part we will deal with introduction to PHP.

What is PHP?

PHP is a server-side scripting language inserted in a common HTML code. What does it mean? Each page containing PHP scripts is taken by the server and all commands listed on the page in PHP, then a clean HTML code is sent to the client (which is a result of the script). Theoretically, the server can look for PHP scripts in all files sent, but mostly it is configured to look for them only in the files with extensions .php, .php3 or .phtml. PHP commands are inserted directly into the HTML code and are separated by tags <? and ?> (or <?php and ?>).

What is PHP good for?

PHP is a very versatile language in which it is relatively easy to programme e.g. a news server or virtual shop. Data can be stored in common text files or in a database (PHP is compatible with almost all commonly used databases, e.g. mySQL). Processing the data from the forms is very easy, various simple online tests including the visitors success rate can be easily created; also a quality advertising system can be easily programmed. The strength of PHP is demonstrated by its use on the servers Email.cz, Centrum.cz or BillBoard.cz

What is necessary to work in PHP?

PHP is inserted into HTML, so any common HTML editor can be used for creating PHP scripts. It is even enough to use Notepad. The most important thing is to be able to place and test the created scripts. For this, PHP support has to be installed in the server (PHP, now in version 4, can be free download from www.php.net). PHP is most efficient

as a module of Linux system, but it can be used also with Windows (you only need this information if you are the administrator of your webpage). If you use any of free webhosting offers, PHP support is not very likely, but e.g. the server www.kgb.cz offers free webhosting with PHP support (with some limitations, e.g. it is not possible to use databases, etc.). In the case of paid webhosting, the provider shall give information about PHP - it is either free or for a certain charge. The provider shall also be able to say what extension it is necessary to use for the files containing PHP scripts (as it was said above, mostly these are .php, .php3 or .phtml).

First script in PHP

Here is the first PHP script that writes the current time:

```
<html>
   <head>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
            <title>PHP – example 1</title>
      </head>

<body bgcolor="#FFFFFF" text="#000000">
            <center><font face="Arial CE, Arial" size="5">
                  Current time: <?php echo Date („H:i:s"); ?>
            </font></center>
      </body>
</html>
```

Instructions: Save in the server e.g. as a file example1.php and watch it in an explorer. For a script to work, it has to be first interpreted by a server, i.e. it could not be explored off-line from a harddisc.

How does the script work?

It shall be noticed that it is basically a classic HTML page that also contains one PHP

command - echo Date („H:i:s“), which is separated from the remaining HTML by tags <?php and ?>. The server first takes the required file example1.php, and seeing its extension is .php, it lets it pass through PHP interpreter and performs all the commands - these are sought between the tags <?php and ?>. It comes across the command echo, which servers for writing into the resulting file. The function Date returns the date and time, in the brackets, there are the parameters determining the format in which the time shall be displayed (hours, minutes, and seconds separated by colons - function Date will be dealt with later). The following semicolon serves to separate the individual commands. It is redundant in this case, as there is only one command, but it is recommended to get used to writing it. Instead of PHP command, the PHP interpreter writes the current time and the server then sends the page to the visitor. If you want to view the source of the script result, you will only see a pure HTML, not PHP. It results that unlike client-side languages (JavaScript), nobody is able to get to your code and thus it cannot be copied.

Summary:

1. PHP commands are inserted in a common HTML code, they are separated by the tags <?php and ?> (or only <? and ?>).
2. To look only for PHP commands, the file has to have the right extension - usually .php, .php3 or .phtml.
3. The visitor will only get a pure HTML code from the server (that is, the code after all PHP commands are performed).

6. Processing http request

6.1. Request methods

GET

It is the most widely used method. It servers for retrieving an object (html file, image,..)from the server. The answer is "cachable". GET request is therefore accompanied by a number of headers in which it is specified, how old the document is, if it has been changed, etc. GET request usually does not have a body.

POST

Using this method, it is possible to deliver the information from the user to the server within a body (POST is often used for sending bigger data from web forms, for uploading files, etc.).

HEAD

is similar to GET, but the body is not transmitted in the response. This request can be used e.g. to find out if the object really exists (for controlling the links within a page).

PUT/DELETE

creates/deletes the given object from the server. These methods are not used very much in practice.

OPTIONS

It is used for retrieving information about a given context (or "*“ for the whole server). The client can find out which requests they can send to the given context.

OPTIONS * HTTP/1.1

Host: www.root.cz

Example of implicit setting of server

TRACE

It is used for tracing the entire request path. In the body of an answer, the client gets the requests of the individual systems through which the request went. This method is used by administrators and web programmes who want to find out why the server is returning an expired document, etc.

Headers

Protocol HTTP version 1.1 defines a large number of headers for requests and responses. There are some of them:

6.1.1. REQUEST HEADERS

Accept*

Headers of this type indicate what the client is able to handle. The server then chooses the most appropriate alternative. The headers include Accept (MIME types of documents, Accept-Charset (character set, very important in the Czech environment), Accept-Encoding (encoding of transmitted data, mostly it is used for choosing a compression), and Accept-Language (language of the document).

Connection

In the protocol HTTP 1.1, parameter "close" is defined, which requires an immediate closing of the connection after the first requested document is transmitted.

Referer

Using this header, the client announces the URL of the page from which the link was generated.

Host

HTTP 1.1 introduces the support of so-called name-based virtual servers. This method enables to operate more virtual servers from one IP address, but the client has to specify the name of the server they want to communicate with.

User-Agent

This header is used to identify the client programme either for statistical purposes or for the purpose of providing different content to different explorers etc.

6.1.2. RESPONSE HEADERS

Content*

Headers describing the content (body) of the response. It can contain e.g. content-length, its MD5 digest (Content-MD5), language (Content-Language), type of the document (Content-Type), and other attributes. It shall be noted that those headers are used not only in the responses. If a body also contains the request (e.g. in the case of the POST method), they have to be used as well.

Server

This header is used to identify the server (usually there is its name, version, and sometimes also other information).

Expires

A server can use this information to indicate the expiration of a document. After this time, the client should download a new version.

There is a number of other headers that can be used e.g. for document downloading

control ("download only if the document has been modified since...") or for provide the server with a user name and password for accessing the non-public parts of the server. Similarly, a server can describe more accurately its response and inform the client when the document was modified for the last time or if its caching in public or private caches is allowed.

6.2. Basic features of HTTP protocol

To fully understand the information, it is necessary to have a knowledge of basic features of HTTP protocol and its functioning. HTTP protocol is an application-level protocol for distributed hypermedia information systems. In practice this means that this protocol is generally used on the internet not only for transmitting data between a client and a server, but for many other purposes. HTTP protocol is stateless, i.e. it does not recognise the clients from whom requests are made. If one client sends a request and then sends another one, the server does not recognise it was the same client.

HTTP exists in 3 versions - 0.9, 1.0 and 1.1 . The first of them, referred to as HTTP/0.9 existed as a simple protocol that could transmit data on the internet in a limited way. Version HTTP/1.0 enabled to transmit data in the MIME format, so it could also contain metainformation about the data transmitted. The most important improvement to the version HTTP/1.1, which is also the latest version, was that all connections became permanent, which means that the connection is closed when either client or server sends a header for closing. HTTP used to close the connection after each server´s response. This improvement considerably accelerated the transmission, because the server does not have to open a new connection for each image, frame and applet.

6.3. HTTP protocol request format

The request of the HTTP protocol has the following format:

METHOD URL OF HTTP DOCUMENT
HEADER
blank line
OTHER-DATA only in the case of the method POST

The request method indicates how the server should process the request. The methods will be dealt with later. Headers are sent in the following format:

NAME OF THE HEADER: HEADER VALUE

Each header has to be in a separate line. All lines have to be finished by tags CRLF (\r\n). At the end of all headers must be a blank line, even if there are no other data.

Requests in PHP are sent through so-called *sockets*. A socket is basically a connection

between a client and a server. To work with them, it is necessary to open the socket first. For opening, the function fsockopen is used:

```
fsockopen(server, port);
```

Example:

```
$sock = fsockopen("www.interval.cz", 80);
```

When the socket is open, it is possible to send the request by means of the function fputs:

```
fputs(socket, request);
```

Example:

```
fputs($sock, "GET /index.html HTTP/1.1\r\nHost: www.interval.cz\r\n\r\n")
```

In the following part, we will deal with HTTP methods.

7. HTTP protocol request methods

In HTTP/1.1, there are seven basic methods of HTTP requests:

- GET
- POST
- HEAD
- OPTIONS
- PUT
- DELETE
- TRACE

After each request, individual headers can follow. In the version HTTP 1.1, in each request it is compulsory to use the header Host, which specifies the host. After headers, a blank line must follow, as it has already been mentioned.

7.1. GET method

The GET method is the simplest and one of the basic method. Every time a page from a server is loaded without sending a form with the POST method first, this method is used for retrieving the page from the server. The result is thus a page and its headers which we ask about using this method. The format of this method is as follows:

GET URL-OF THE PAGE OF THE PROTOCOL VERSION
HEADERS
blank line
Example:
GET /index.asp HTTP/1.1
Host: www.interval.cz
blank line

7.2. POST method

The POST method works as the GET method, but in the case of the POST method it is possible to send data to the script after the headers and blank line. This method is used for sending the data from a form with the POST method. The format of this request is as follows:

POST URL-OF THE PAGE OF THE PROTOCOL VERSION
HEADERS
blank line
DATA FROM THE FORM

Example:
POST /processdata.php HTTP/1.1
Host: www.formulare.cz
Content-Length: 29
Content-Type: application/x-www-form-urlencoded
blank line
array1=value1&array2=value2

In the case of this method, there are mroe headers that have not been mentioned yet. Content-Length indicates the length of the data from the form (in bytes), and Content-Type: application/x-www-form-urlencoded designates the MIME type of the data from the form.

HEAD, OPTIONS, PUT, DELETE, TRACE

If you do not programme an Internet explorer, you are not likely to encounter these methods. Therefore, the following table shows only the basic meaning of these methods.

Method	Description
HEAD	It works as GET, but it does not return the body of the page, only the header. This is used e.g. for finding out whether the page has changed from the last request.
OPTIONS	It is used for requests about server possibilities
PUT	It words as GET, but it retains the body of the request at a location given by the required URL. It is similar to sending files via FTP.
DELETE	It removes the document from the server. The document that is to be removed is given the URL of the request.
TRACE	It is used for tracing the request through all proxy servers and firewalls the request passes through. It is similar to the tool TraceRoute.

7.3. Using data sources

Modern and efficient data processing applications do not store information directly in their own files, but they use some of the external data sources. There are a number of data sources, starting from a simple text file through file databases e.g. in the DBF format to databases stored in SQL servers.

Each such source usually uses its own format for storing data and methods for retrieving the data stored. To avoid the problem with the diversity of data sources and to avoid the necessity to develop applications for each of such sources, applications programmers created a standard tool that enables to access various data sources by means of one standardized platform – ODBC.

Such a standardized platform enables the users to use the data from various sources at various locations in a complex and simple way. For example, it is possible to use the data from Access in Word or Excel without having to export them and load them into the required application first

7.4. What is ODBC

ODBC stands for Open Database Connectivity. ODBC data sources are accessible to applications through a relevant driver which can be perceived as a mediator for the communication between a user´s application and external data source. The application sends the request about the data to ODBC. The relevant driver translates the request so that the external data source understands it and sends it to the data source. The response from the data source is also sent through ODBC, which translates the result into the standard form and returns it to the application.

The principle of ODBC is standard; therefore any application capable of using the ODBC driver can access any external data source from any producer that provides the appropriate driver. ODBC thus enabled to standardize the access to the data for the applications without solving how the data source works.

On the one hand, the application only needs to know how to work with ODBC. On the other hand, the producers of various data sources provide the relevant ODBC drivers so that their sources are easy to access for the applications. This is advantageous for the applications programmers that have independent standardized access to data, as well as for the users who get the applications that can access various data in a cheaper way and faster.

Working with ODBC in Windows

ODBC setup controls are already an integral part of the operating system Microsoft Windows. **ODBC data source administrator** is in the control panel. Here it should be not-

ed that the procedures and images are valid in the operating system Windows 10, but similarly they are valid in other versions of Windows. However, in some operating systems, the ODBC administrator may be accessed in a different way.

8. Data sources

To fill the mailing lists and templates with the right data at the right time, Mailkit uses **XML & RSS data sources.** Their use is limited depending on the type of user account. The version Mailkit Base is limited to using XML data sources intended only for the import of the recipients into lists, while Mailkit Syndicate and Agency support both XML and RSS data sources even for loading data in templates.

Using XML data sources for recipients list

To create a new list of recipients from a data source it is necessary to set up a new XML data source.

- Name - name of data source. If the data source will work for the recipients lists, the list will have the same name.
- Description - description of data source.
- Source - URL address of the XML or RSS which will work as a data source.
- Authorization - it is ticked if the access to the data source location is password protected.
- Type - you can choose from RSS or XML data source.
- Destination - the data source destination is selected from the drop-down list. It will be used either for the list of recipients or template.
- Update automatically - if this option is ticked, the source will automatically update before the campaign is sent.
- Expiration time - it sets up the time after which the data source shall be update in the case of automatic updating.
- Last update - it shows the date of the latest data source update
- Empty records:
  Reset - the recipients data will be deleted according to the empty records in the import.
  Retain current value - the recipient data will remain in the same form as before the import.

How to prepare data source

The data source has no defined structure and can be in the formats XML or JSON, which must be fully valid. The data source file must be posted to a URL available from the Mailkit servers and secured against third-party access, as it is sensitive information.

As each client uses different information system with different possibilities, the data source system is universal and does not set specifically the structure of the required data. However, ther are some technical limitations of the data sources:

- Fully valid XML or JSON
- Attributes are not supported in XML format (e.g. first_name="Jana" gender="f" country="cz")
- UTF8 character encoding recommended
- Do not use "," for separating more values - use "|"
- The unique record identifier and the only compulsory array is the e-mail address. If there are more records with the same e-mail address, they will be overwritten.

As mentioned above, the only data compulsory is **e-mail;** other data are not compulsory, but important. The general rule is "the more, the better", but also nothing must be exaggerated. The data source shall be important the maximum of the recipient data available which are possible to be used for existing and future e-mail campaigns.

Import of data from data source

To obtain content, the source branches must be assigned to the contact fields. To assign, click on the name of the data source and click on the **View Structure** button. After assigning all fields, click on **Save**. Subsequently, it is possible to continue with **Import** for the current data import. At this moment, a new **Recipients list** will be created (it will have the same name as the source that created it), and the data from the data source will be imported according to the previous assignment.

At the same time, Syndicate and Agency account users have a possibility of automatic update. If automatic update is chosen, the data source will be automatically imported and the list of recipients will be updated before each campaign delivery begins. This parameter is not recommended for data sources containing more than a thousand of records, since the update can take a few minutes, which delays the delivery of the campaign. For larger data sources, it is recommended to use the possibility of regular planned update scheduled for a specified time of day by the customer support.

Using XML & RSS data sources in templates

Setting up XML & RSS data sources for the use in templates is similar to the use for recipients lists, but without having to assign the meanings to each field. The values are determined by a string of names in the template, therefore it is easy to set up any XML or RSS source.

Above, there is an example of a template code for which the RSS data source called *EXAMPLE* is used. The command *FOREACH* creates a loop for parsing and finding all records. Each of standard RSS tags is easy to solve and insert into HTML code, allowing the data output in the template. For more information, see Emails templates.

Product data sources

Data sources can also be used for transmitting the product offer to Mailkit and for subsequent use of product information in campaigns. This is where the power of data

sources and programmable templates are shown, which enables to combine data from more sources and personalize the content for the individual recipients automatically.

For product information, it is possible to use any of the common product feed formats for Heureka, Zbozi, Google, etc., or generate the own feed with the necessary information. Since the product feeds are very extensive and the speed of working with the data contained is important, these data sources are transmitted directly into the SQL databases, and it is still possible to work with them. For setting the product data source, you shall contact the customer support that will help you with the implementation and use.

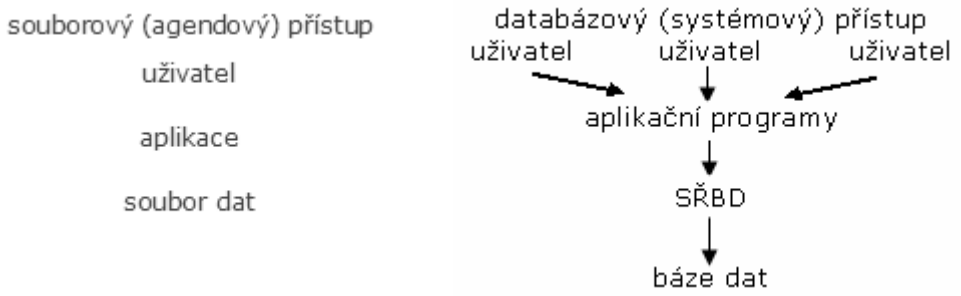
Delivery feeds

Delivery feeds are special data sources that are used for passing structured information for the realization of the campaign delivery. While a campaign usually uses the pre-set list of recipients, which is used for delivering the campaign according to the rules set, in the case of delivery feed, the campaign is sent only to the addresses listed in the feed. It is an alternative to API call mailkit.sendmail_mass, that is, the way to get highly structured data to be processed when sending the campaign to Mailkit e.g. from personalization systems or CRM. These feeds must have a strictly defined structure in the XML format.

9. Database approach

Requirements to database system:

- data consistency check - database shall be able to ensure compliance with certain rules of so-called integrity constraints and secure ensure data against possible accidents that may occur during transactions.
- transaction - sequence of manipulation with data, which must take place in order to ensure that the data are stored properly, e.g. transfer from 1 account to another account (this must be done correctly in both accounts)
- large volumes of data - relatively to the possibilities of storage media, the database must be able to hold adequate amount of data
- data management - development stages



Legend: souborový (agendový) přístup - file (agenda) access, uživatel - user, aplikace - application, soubor dat - data file, databázový (systémový) přístup - database (system) access, aplikační programy - application programmes, báze dat - data base

Database access

- large database system - companies Informix, Sybase; smaller (more affordable) database systems - MS Access, Paradox, FoxPro; and small database systems available for free - e.g. My SQL
- SQL language- standard that enables to use data sources managed by various database systems

Creation of a database for the IS organisations - a complex issue requiring people with different professional skills. When designing the conceptual scheme of the database, it is decided what the database will contain. During the IS operation, it is important for the users, if they are able to use the data base as an information source.

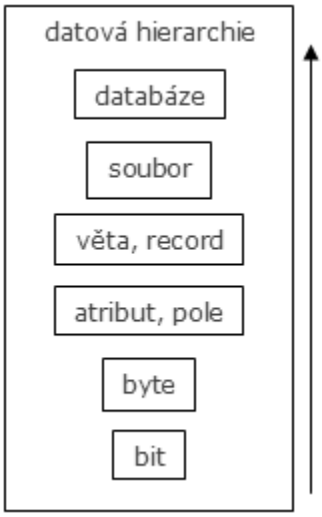
Requirements to computer IS:

- horizontal and vertical integration of information
- fast aggregation of information from the lower stages
- rational presentation of the information in time, form, and space
- time frequency

- Extent of the information stored

Data organization - files and databases

- transactions processing: batch processing or online processing
- current trend - object-oriented and hypermedia databases



Legend: datová hierarchie - data hierarchy, databáze - database, soubor - file, věta, record - statement, record, atribut, pole - attribute, array

Design of structured database

- the reality whose reflection shall be the designed database. It consists of several objects (entities)
- there can be different relations between the monitored entities (e.g. between the entities of the same type = recursive relationship)

Cardinality of the relationship: symbolic designation 1:1 (professor XY´s wife is ZN); 1:N (professor XY teachers students); M:N (which students are taught by which professors)

Integrity constraints of the database - all rules defining the allowed values (a combination of values) of the attributes, display format

Relational data model

- It assumes the existence of single-value attributes
- representation in the form of a relational table, in which a tuple corresponds to a row, and an attribute corresponds to a column

Relational database

- all data is in the form of 1 or more tables named by columns
- each column contains data from 1 domain (i.e. 1 data type)
- the elements of the individual columns (with name and type) are usually called

items or arrays, and the term row corresponds to record (statement)
- in terms of a relational model, sessions describe both entities and the relationships between them
- based on this dual use, it is possible to distinguish between so-called entity sessions (sets of arranged tuple attributes describing the entities) and relationship sessions (sets of arranged tuples).

Data warehouses

Warehouse principle - 2 main objectives

- unification of data view in the individual so-called production systems provides a clear access to data - differently called but the same objects are seen as one object
- differently measured but the same variables are measured by the same measuring tool

10. Structured and unstructured data

10.1. Structured data

Basic types (classification to distinguish allowed and non-allowed manipulation and values)

- text (strings, expressing information by means a text code, certain set of elements that can be recorded, or that can define syntax)
- numerical - real rational numbers
- date, time - limited values it can achieve (February 30, 27:00)
- logical - meeting the conditions of existence or non-existence of an object characteristics - 2 values (0 and 1, A and N)
- category - value of characteristics selected from a scale (often dials, enables to record values only by means of a code)
- Structuring creates a data organisation that enables efficient storage, processing and retrieval of data as needed → structured data create search keys (sometimes referred to as identification keys) - keys that clearly identify data records are called private keys (identification keys) - a basic condition of data and database system
- Information about something (reflection of reality) - Name and surname, address, age, phone number, weight, price,... number of points, category, average grade,... number of pieces, number of pages
- Operations or what can be done - addition, rounding, multiplication, connection (name+surname), shortening, sorting,..., day in a week, negation, ...

- Data type (must be defined) - number, text, date and time, logical data (yes/no)
- Encoded data - various encodings - text, letters - various code tables (ascii, ebdic, ...) national alphabet; date and time (how we write a date)

10.2. Unstructured data

Data type: free text, audio, video, graphics, multimedia

- Provide more information that structured data
- Problem: it is very difficult to search using unstructured data. Solution: unstructured data are supplemented with structured data (mp3 title)

10.3. Data volumes - structured and unstructured data

- ascii text page (notepad) 1.8 kB
- word text page 50 kB
- vector graphic A4 30 kB
- bitmap image A4(jpg, rgb) 5 MB
- 1-minute audio record (wav) 10 MB
- 90-min video record 3 GB

Data Warehouse (DWH) is a special type of relational database, which enables to solve tasks focused primarily on analytical querying over large data sets.

Subject orientation

In the case of a common relational database, it is common to try to minimize the redundancy of data stored, which is achieved by their normalization into third normal form and internal interconnection of the individual logical functional units. In the data warehouse, there is always an effort for a clear internal separation of the individual functional units. The result is a structure clearer for a user (manager, business analyst) at the expense of increased requirements for a storage space.

Integration

A common operational application (a programme) over a relational database solves a certain specific kind of tasks over "its" specific data. In a data warehouse, it is necessary to gather the information from many various sources and group them not by their origin but by their logical meaning (it is closely related with the subject orientation - all data concerning a certain functional area must be "on one stack" regardless where they are from).

Low variability

The data are usually uploaded into a warehouse in larger batches (e.g. at weekly or daily intervals) and are not modified afterwards.

Historicization

The data in a data warehouse are usually stored in their historical form, not only in its current state. This is due to the necessity to carry out analyses focused on the development over time. From the users´ point of view, in a common relational database, only the current state of data objects is interesting.

10.4. Technological characteristics of data warehouses

The requirements for data warehouses imply their technological characteristics:

- A data warehouse must contain a tool for uploading data from various data sources. These sources can have different data formats and different physical location. They do not have to be only relational databases.
- A data warehouse stores data not with regard to the best editing conditions, but with regard to the best and fasters execution of complex requests. Therefore, for data storing, OLAP technology is often used.
- It cannot be said which requests and tasks will the users have in the future (at the time of creating a data warehouse, we only know the type of the tasks, not the individual tasks and requests).

10.5. Logical structure of data warehouse

From the logical (user´s) point of view, the data in a data warehouse are divided into diagrams. Each diagram corresponds to one analysed functional area.

The core of each diagram consists of one or several fact tables. They store the analysed data - numerical and financial values which are used for analytic calculations - aggregation, sorting, etc. Most storage space in a data warehouse is occupied by fact tables containing detailed information from all sources, i.e. more than other tables.

Fact tables are linked to dimensions by means of foreign keys. Dimensions are tables that contain lists of values used to categorize and sort data in fact tables.

Example

It is necessary to store information about all sales from cash registers of hypermarkets in a data warehouse. The data will be further analysed on the basis of the time of the sale, shop, type of goods, supplier, ongoing marketing events, and the payment method (with a card, in cash).

The Sales diagram will contain a fact table - Sales items, where information about the

type of the goods sold, price, and number of pieces (or the amount sold) will be stored for each item sold.

Besides this fact table, the diagram will also contain dimensions for sorting the sales items: time dimensions Date and Hour (within a day), dimensions Shop, Type of goods, where there is one row for each item (e.g. Strawberry yoghurt 250 ml), Category of goods (Yogurt), Department (Dairy products), Supplier (Dairy XX), etc.

The fact table must be directly or indirectly linked to each of these dimensions using a foreign key.

There are two options in terms of storing hierarchical dimensions:

- From the entire hierarchy, one dimension table will be created that will store the data for higher levels of the hierarchy redundantly. This way a diagram is created where each dimension table is linked directly to a fact table. According to the shape, this diagram is called a star schema.
- 3NF will be applied to hierarchical dimension, so only the dimension at the lowest level of the hierarchy will be linked directly to the fact table. Other dimensions will be linked to any of the lower dimensions in the hierarchical structure. This is called a SnowFlake schema.

11. AJAX

AJAX stands for Asynchronous JavaScript and XML. It is a generic term for technologies of the development of interactive web applications that change the content of their pages without having to completely upload them. It is performed using asynchronnous processing of web pages by means of a library in JavaScript. Unlike classic web applications, they are more user-friendly, but require using modern web browsers.

These applications are developed using the following technologies:

- HTML (or XHTML) and CSS for presenting information;
- DOM and JavaScript for displaying and dynamic changes of presented information;
- XMLHttpRequest for asynchronnous data exchange with a web server (typically, XML format is used, but it is possible to use other formats, including HTML, text, JSON or EBML).
- Like DHTML, LAMP or SPA, Ajax is not a concrete independent technology. It is a term referring to several technologies together with a specific goal.

Advantages

Advantages include elimination of the necessity to reload and redraw the entire page in each operation (unlike the classic model of www pages). If a user clicks a vote button in an opinion poll, the whole page must be reloaded from the server, even if only the vote results are updated and the rest of the content remains the same. Using AJAX, the vote is sent in the background, the server sends only the parts of the pages that have changed, and those parts are updated and redrawn on the page. Moreover, there is no unpleasant effect when after the activity, its block elements, images, etc. are gradually adjusted and formated in the continuously uploaded page. It can also be annoying that after the specific activity, in the middle of a longer page (scrolled down), the newly uploaded page is displayed scrolled up. Using AJAX, the user works more fluently and the speed (especially in the case of faster internet connection) is nearly the same as the speed of common desktop applications.

This also has the potential to reduce the load on the web servers and web in general. Since it is not necessary to create the whole HTML document for each request, but only the changes made have to be marked, the volume of the data exchanged is significantly smaller and theoretically, it can also positively influence the load of the database servers or other backend systems.

Disadvantages

On the other hand, AJAX can increase the *number* of exchanged HTTP requests, and even though they transmit a smaller volume of date, in the case of inappropriate implementation the load does not decrease.

The disadvantages include mainly the changes in the web usage paradigm: web pages behave as applications with a complex internal logic, not as a sequence of pages which can also be navigated using Back and Next buttons. Similarly, it is not possible to pass

the URL of the page in which something has been "clicked" using the AJAX technology. Modern AJAX applications are able to recover (at least partly) history browsing using various techniques (e.g. using a part of the address after the # or using invisible IFRAMEs). This, however, makes the pages design complicated and more time-consuming, as well as more work on implementing using AJAX.

Another problem of AJAX applications can be the network latency: the need for Internet communication has a negative impact on the response speed and the user interface interactivity. If the user is not announced clearly that the application is processing their request (and communicates with the server in the background), all they register is a delayed response (the users may even try to start the operation again as they think that the system has ignored the request, thus generating higher load to the server and / or cause something they did not plan, e.g. to order ten tickets instead of two). As a suitable solution, it is recommended to show somehow that the user´s request is being processed, e.g. using text or animated icon.

Another AJAX disadvantage is the necessity to use modern graphics browser that support the necessary technologies. However, all currently used common browsers support these technologies at least basically; the problem is only with the minority Lynx browsers or with hardware-weaker browsers, e.g. in some mobile phones and PDA. Within web accessibility, it is required that the pages are accessible from browsers even without AJAX, which means more time and work for the developers and higher costs for the site buyers.

AJAX

As it has been mentioned above, AJAX stands for Asynchronous JavaScript and XML. AJAX is a modern technology frequently used in the current web applications. It is often mentioned, as it is a part of RIA, a new programming style leading to higher user´s comfort and functionality of applications.

AJAX is not a new technology, it is only a combination of already known technologies - HTML (or XHTML), JavaScript, XML and XMLHttpRequest.

Why is AJAX so advantageous? The applications using AJAX can send and retrieve data from a server without having to reload the whole page (unlike classic links). AJAX can thus be used for various purposes, e.g. autocompletes (forms that fill in automatically depending on the key pressed), AJAX opinion polls, and other more complex applications that are able to facilitate the work of a user.

AJAX has also several disadvantages, especially when used improperly, it significantly reduces the usability of the pages. As with other technologies, it is therefore necessary to plan AJAX application carefully and test it on users.

As you already know, Javascript is so-called client-side language, which means that it is done on the client´s web browser side and is interpreted by javascript interpreter. It has a wide range of characteristics which include also the capability of performing asynchronnous operations / tasks.

What exactly does AJAX asynchrony consist in? It is the ability of Javascript to call a script

or element API on the server and not wait for the response. Instead, code execution continues (for example, the user can see the page and can work with it). When the response comes, code context execution is stopped (sooner or later, depending on the priority of the task) and there is a callback (it is a function that is performed if the server provides a response). Such a code or function is called "non-blocking", because it does not block the course of script by waiting for the response or processing.

AJAX is a method of programming in Javascript. It is not a new language or framework or a third party library. It is a method of exchanging the data in the application with a database, server scripts (PHP, Java, ASP, etc.) without having to update / reload the whole page. Technically, we do not update (in terms of refreshing) any part of the application or a page - this is actually the AJAX essence.

Why XML is contained? It is one of the data format in which AJAX can retrieve the information from the server. The most widely known and used are JSON, XML, text, binary data. Each of them can be used in a different way.

AJAX enables to check the data entered by the user even before the user sends them by confirming the form. To some extent, this can also be done by Javascript, but on in the case of a registration form with a conditional password, user name or e-mail, which have to be unique within a table or database, and these are exactly the situations for which AJAX is the right technology. It enables to perform the same operations that could be performed by calling PHP scripts when redirecting after sending the form, but unlike them, it allows you to make this call in the background.

AJAX basics

The main idea behind AJAX is in the fact that parts of a web page are uploaded asynchronously, thus changing the content of the page. It means that changing the content will not cause uploading of the whole page, only the modified content. For this purpose, the object XMLHttpRequest is called from JavaScript. This request asks the server about the content that is typically rolled back using XML and JavaScript. The script then analyses this XML. The data from XML script then translates and changes the web page using DOM (Document Object Model) correspondingly.

The basic principles of working with AJAX will be described on a practical example - a discussion forum, in which the discussions in HTML are displayed as a tree. If the users want to read a contribution, they have to click on its headline. In the case of a classic model, the request would be send to the server and subsequently a whole page would be created, that would contain the text of the contribution and the whole "tree". In the case of AJAX model, only the text of the contribution will be sent as a reaction to the request. Adding the text to the original web page will be ensured by a relevant JavaScript programme.

An ActiveX object was available for older Internet Explorer versions (starting with the version 5.0) for performing this task. Like other explorers, IEZ makes the corresponding object available natively. The current API XMLHttpRequest of the Internet Explorer is available on MSDN; Mozilla project API is also available.

AJAX´s own trick is XMLHttpRequest – it can be processed asynchronously, which in this context means asynchronously to the arrangement of the rest of the page as well as to the running scripts. This means that the data can be downloaded by means of HTTP and that the data can be downloaded without affecting the interaction of the user and the page. The requested page is also created in the background.

What is AJAX? Simply said, AJAX is a technology that uses scripts to facilitate the communication of a page and a web server. This way it is possible to change the page content without having to upload it again.

Theoretically, it is possible to create dynamic pages without server scripts, that is, to have one page for the whole web, which only changes its content. However, this is not recommendable - as it has already been said, it is a script technology, so it depends on the client and it cannot be secured it will work. Nevertheless, it can be used as an add-on for web pages - e.g. opinion poll or a search assistant as in the case of Seznam.cz.

If you are sure that AJAX will work for the visitors of your web pages, it can be used also for a more sophisticated applications. It can be used e.g. for a chat that downloads only new contribution; if the visitor is alone there, checking for updates slows down, which saves a lot of data transmitted.

12. Frameworks

Framework (application framework) is a software structure, which servers as a support for programming and development and organization of other software projects. It can contain supporting programmes, API libraries, support for design patterns or recommended procedures in the development.

- Purpose
- Architecturey
- Examples
- Related articles
- External links

Purpose

The objective of a framework is to take over typical problems of a given area, thereby facilitating the development so that the designers and developers can focus on their tasks. For example, the team that uses Apache Struts for developing web pages for a bank can focus on bank operations and not also on ensuring a perfect navigation between the individual pages.

It is argued that using frameworks will make the code slow or inefficient and that the time saved by using a foreign code has to be given to studying the framework. However, its re-deployment or its use for a large project saves a significant amount of time. After uninstalling the framework, some applications will not be able to work any more.

Architecture

Frameworks consist of so-called frozen spots and hot spots. Frozen spots define the overall architecture of the software structure, its basic components and the relations between them. These parts do not change in any framework use. On the contrary, hot spots are components that create a completely specific functionality together with the programmer´s code; therefore they are each time different.

In an object-oriented environment, framework consists of abstract and classic (non-abstract) classes. Frozen spots can be represented by abstract classes and teh code (hot spots) itself is added by implementing abstract methods.

Framework is a software structure that serves as a support in programming and development and organization of other software projects. It can include supporting programes, API library, design patterns or recommended procedures in the development.

Another very important factor in choosing is what the framework is needed for and its purpose. Generally, frameworks can be divided into two groups:

- Sets of scripts – libraries – covering all kinds of needs
- Scripts creating one concrete web application

Advantages

- Faster development

- Less code
- Universal code – changes or new functions are added in a much easier way
- Nice URL

The development speed is higher. Framework makes working easier, avoiding programming routines such as connecting to a database, checking the right option with a sect, or validating a form. Cool-Url, XSS, separate application and presentation logic, changing DB server from MySQL to PostgreSQL will be really easy.

Thanks to framework, it is possible to focus only on the development, but it is redeemed by the time you need for learning to work with frameworks. When you learn to do something, soon you will find out it can be done much easier and faster. Learning to work with framework can be a matter of a month, but also a matter of several months or a year. Finally, you will extend the framework by libraries and you will be able to significantly shorten the programming time. Framework means saving time and money

Choosing a framework is not easy. It is recommended to have at least two - one for easy and second one for more complex tasks. Before each project, it has to be decided which one is more suitable. Of course, the best thing to do is to learn those with a good large community, those that you are sure their development will not be finished in a week.

- CakePHP: quite easy to learn; quality community; long development of a new version
- Zend Framework: very efficient; it covers all needs in creating any web applications; large community; sophisticated; long titles;
- CodeIgniter: small, simple; developed by a company, not by a community; better than Cake in some issues x "stupid" in others (it does not have layouts)

Monolithic frameworks gradually decouple into separate components, which brings a lot of advantages. While using only one framework component used to be difficult and sometimes even impossible, nowadays it is possible to install its component easily. The development cycle of individual components is different. They can have their own repositories, issue trackers, development teams.

Components can be updated no new versions continuously, without waiting for another version of the whole framework. It is also possible not to update some component, e.g. because of a BC break.

The meaning of the term “framework” is thus shifting, and ”versions” now almost do not have their original meaning. Instead of a framework XYZ, version 2.3.1., a set of component in different versions working together is used.

12.1. What is a framework?

When creating modern web applications, frameworks, or development frames are often used. It is a set of libraries and source code that could be repeatedly used in order to facilitate work and whose functionality can cover a part of the application created. Frameworks can solve problems that can be generalized in some way; otherwise creat-

ing a framework would not make a sense. This allows the application creator to focus on the functions that are unique and exclusive for the application, and does not have to solve routine problems.

Using Javascript frameworks

Frameworks can be created for most programming languages. Javascript framework is thus a framework that is used to facilitate work and programming in Javascript. Frameworks use is very wide and every day, the portfolio of problems Javascript frameworks are able to solve is bigger. Generally, it can be said that they can be used for effective writing of source code; to a large extent they also solve incompatibility between web browsers and a written code, thus saving the programmer´s time and enabling to use elements that would be difficult to programme. Their strength and use is mainly connected with using the AJAX technology, as they improve the interaction of the user and application by means of asynchronous communication between the client and the server. Furthermore, they enable dynamic and simple access and change of the individual elements of a page (DOM model), GUI elements, assign events and animate them. Last but not least, some frameworks contain pre-built GUI components that either cannot be implemented using any other technology or it is too difficult and inefficient. These elements are easy to work with and implement within a framework using just several code lines.

13.10 Best PHP frameworks for developers

PHP, known all over the world as the most popular server-side scripting language, has developed a lot since in static HTML files, first inline code fragments started to appear.

At that time, developers had to create complex webs and web applications; at a certain complexity, it was too **time-consuming and it took too much effort** always start from scratch. Hence the need for a more structured and natural way of development appeared. PHP frameworks provide an adequate solution to this problem for developers.

Why to use PHP framework

Firstly, we will deal with the strongest reasons why many developers prefer using PHP frameworks, and how these frameworks improve the development process. PHP frameworks advantages are as follows:

- Enabling rapid development
- Providing well-organized, reusable and maintainable code
- Enabling growth over time, since framework-based web applications are scalable
- No worries about low-level web security
- Following MVC (Model-View-Controller) pattern, which secures the separation of presentation and logic
- Encouraging modern web development practices including object-oriented programming tools

1. Laravel

Although Laravel is a relatively new PHP framework (it was released in 2011), according to the latest online survey on Sitepoint, it is the most popular framework among developers. Laravel has a huge ecosystem with a platform ready to be hosted and deployed immediately, and its official web offers many tutorials in the form of screencasts called Laracasts.

2. Symfony

The components of the framework Symfony 2 are used by many projects, such as the system for managing the content of Drupal or the software phpBB for running forums. It is also used by Laravel. Symfony has an extensive developers´ community and many supporters.

3. CodeIgniter

CodeIgniter is almost 10-year-old lightweight PHP framework (originally released in 2006). CodeIgniter has a very straightforward installation process that requires only minimal configuration. It is an ideal option to **avoid conflicts with PHP versions**, since **it works smoothly on almost all shared and dedicated hosting platforms** (currently, it requires only PHP 5.2.4).

4. Yii 2

Choosing Yii frameworks gives a page a boost for performance, since it is **faster than**

other PHP frameworks. It extensively uses the **"lazy loading"** technology. Yii 2 is purely **object-oriented** and it is based on the **DRY** (Don't Repeat Yourself) coding concept, so it provides a **clear and logical code base.**

5. Phalcon

Phalcon framework was released in 2012 and soon became popular among the PHP developers. It was said to be as fast as a falcon, as it **was written in C and C++ languages in order to achieve the highest possible performance optimization.** For using Phalcon, it is not necessary to learn language C, because the **functionality is exposed as PHP classes ready to be used in any application**

6. CakePHP

CakePHP has been used for a decade (first version was released in 2005), but it is still one of the most popular PHP frameworks, since it has always strived for being up-to-date. The latest version, CakePHP 3.0, enganced session management, improved modularity by separating several components, and increased the ability to create **more self-sufficient libraries**.

7. Zend Framework

Zend is a robust and stable PHP framework packed with a number of configuration options; therefore it is usually **not recommended for smaller projects.** It is, however, great for **the more complex ones.** Zend partners include e.g. IBM, Microsoft, Google, and Adobe. The coming new version, Zend Framework 3, will optimized for PHP 7, but it will still support PHP 5.5.

8. Slim

Slim is a PHP micro framework that provides only what is needed. Micro frameworks´ design is minimalist, they are great **for smaller applications**, for which a fully equipped framework would be too much. The Slim creator was inspired by a micro framework Ruby called Sinatra.

9. FuelPHP

FuelPHP is a flexible full-featured PHP framework that supports not only MVC, but also its developed version, HMVC (Hierarchical Model-View-Controller). FuelPHP adds an **optional class** called Presenter (previously called ViewModel) between the View and Controller layers **to include logic necessary for generating views**.

10. PHPixie

PHPixie is a brand new framework started in 2012 in order to create a high-performance framework for read-only webs. Like FuelPHP, PHPixie **implements design pattern HMC** and is created by means of independent components, that **could be used also without the framework.** PHPixies components are 100% tested through unit tests and require only a minimum of dependencies.

WEB TECHNOLOGIES

1. HTML

HTML or **HyperText Mark-up Language** is a tool for creating web pages - the current version - HTML5 (HTML 5.2). This language is standardized by the W3C RGANISATION s.

Consists of tag (s) and attribute (s).

HTML viewing takes place using a browser, that is in several steps (= parsing).

- The browser retrieves the document and performs DTD syntax analysis
- Each element is assigned a style (display mode)
- Apply script scripting code
- Step-by-step rendering of the page

We design HTML tags by relevance to:

- Structural structures that help structure the document. Example: Paragraph (< p >), headings (< h1 > , < h2 > ...) ...
- Descriptive (semantic), describing the nature of the content of the element , eg: Title (< title >), address (< address >) ...
- Stylistic , which determines the appearance of an element when viewed, for example: Bold (< b >), italic (< i >), highlight (< strong >) ...

Each HTML document should follow the **basic schema**:

At the beginning is the DTD directive, the document type declaration:

```
<! DOCTYPE html >
```

Follows to the element:

```
< html > </ html >
```

A document's desk (Includes meta data):

```
    < head >

        <meta charset = "utf-8"> - Encoding

        < title > Page title </ title >
```

Author, description, keywords, cascading styles...

```
</ head >
```

Behind the header follows the body of the document, which summarizes the content of the page as text, images, links, tables...

```
<body> </ body>
```

To write HTML, it is advisable to use a text editor that guides the colour syntax (colour coding for individual parts of the code - tags, properties, plain text) , d telling the marker, knowing the tabs, or managing to validate the document according to the prescribed specification.

For example: Notepad ++, PSPad...

Alternatively, you can use WISIWYG (What You See Is What You Get) editors that directly link to the finished page . The user does not need to know HTML - folds and page editor generates the appropriate code.

For example: Adobe Dreamweaver, Microsoft Expression Web, LibreOffice Writer / Web, Bluegrifon

2. CSS - Cascading Styles

CSS is a language for describing how elements are displayed on pages written in HTML, XHTML, or XML, developed and standardized by W3C. The current version is CSS 3.

CSS can be connected to HTML code in several ways:

- In html page code using element u < style > </ style>

```
<style = "text/css ">
   # head {
                width : 200px;
                height : 450px;
   }
</style>
```

- Help with external file - element <link>

```
<head>
        <link rel = 'style sheet' href = 'styles.css'
        type = 'text/css '>
<head>
```

- Direct inline style enrolment using the attribute style

```
<p style = " color : red ; text- decoration : underline ">
This paragraph will be red and underlined. </ p>
```

PHP language is composed of Rules where each rule contains a selector and a block of declarations. Each block of declarations then contains individual declarations separated by a semicolon. Each declaration consists of property identifier, colons, and values. It can still be tagged! importatn , which increases the force of the declaration.

Why use CSS instead of HTML formatting?

CSS offers:

- more formatting options that HTML does not offer - for example : determining the distance of elements from the page margins .
- Easier to edit your appearance - change the colour of all headlines at once, change the font, ...
- Possibility to create a template for multiple pages at once
- Separation of structure and style - in HTML content, in CSS appearance
- Caching styles - Faster page retrieval (but extra one HTTP request to load an external file)
- Dynamic changes to CSS properties using JavaScript
- With CSS, any XML language can be formatted
- Display settings for individual devices - conditional CSS

The end user can write his own CSS style for any page - you can set all the links on each site to be always underlined, or that the font is always black on that particular site.

Combined with JavaScript, effective bookmarks can be created that can improve the appearance of the page. For example, remove all background images; change the background to white and the font to black, etc.

WITH CSS also have some drawbacks. The main thing is that there is always enough support in most browsers or bugs in implementation of CSS in browsers. This can be done by using different styles for different browsers.

Conditional comments can be used to set different browsers

```
<! - [ if IE]> <style = ”text / css ”> #
alert { color : blue; } </ style> <! [ endif ] ->
```

This code will only be interpreted by Internet Explorer, other browsers will see a common HTML comment and will not interpret the internal style sheet.

I CSS has some limits.

- CSS selectors do not provide access to parent elements
- For example, you cannot just strike those paragraphs that contain a link.
- Horizontal control of the elements on the page is intuitive and simple, while vertical styling requires a more comprehensive approach (eg: flexbox or grid).
- CSS does not provide the option to symbolically write variables or constants
- All values must be written directly into the code.
    - For example, if the same colour is used in multiple locations, symbolic entry color = red; and then just write the colour variable, the red value must be entered everywhere. This limitation removes CSS pre-processors (like SASS, LESS, Stylus).
- CSS2 does not offer any option for creating circular frames or other round objects. Works only with rectangles.
- CSS2 does not offer any option to assign multiple background images to one element.

3. CSS II

You need to clarify what CSS offers and defines.

CSS Version 1 introduces the syntax used in i The following versions offer a way to select elements through selectors, multiple pseudo-classes, and define values and their units.

Categories for which CSS 1 defines values and units:

- font properties
- text colours and backgrounds
- text properties
- properties of block elements
- how to display elements
- position management

Values and units:

A decimal point is used for decimal values

Length units

- (none) - for dimensionless properties (e.g., line- height)
- % - Percent, unit relative to the implicit dimension, written without space
- pt - typographic point, default unit is 1/72 inch
- px - 0.75 pt
- PC - pica, 1 PC = 12 pt
- cm - centimeter
- mm - millimeter
- in -
- em - square , is equal to the base font height
- ex - the height of the letter "x" is relative to the font used

Colors in CSS are entered using the RGB palette and this:

- Either numerically # rrggbb - two-digit hexadecimal value (00 to ff) - 16 million colours
- # rgb - Hexadecimal single-digit (0 to f) - 4,096 colours.
- As a safe colour, which is an even smaller subset of colours: These are single-digit colours whose values increase by 3 - combinations of six values {0, 3, 6, 9, C, F} = 216 colours.
- rgb (r, g, b) - in decimal (0 to 255) and syntax as a function - 16 million colours
- rgb (r%, g%, b %) - values in the range (0 to 100)

Alternatively, you can use predefined constants text based names named colours. E.g. aqua (bright blue-green), black (black) = # 000, Blue, fuchsia (aniline red), gray, green, lime, maroon, navy, olive, purple, red, silver , teal , white (white) = #FFF, yellow. In CSS version 4.0, 148 colours are named.

Font in CSS 1 has its attributes.

- font-style: normal, italic (italic), oblique (inclined roman)

- font- size : medium; smaller sizes: xx-small , x- small , small ; bigger size: large , x- large , xx-large , smaller , larger , size in percent 100% is the standard size
- font- weight: boldness: normal, bold, bolder, thicker, you can enter the number: 100, 200 ... 900 (400 = normal, 700 = bold)
- font-variant: small-caps , normal
- font- decoration: underline (underlined), overline (overlined), blink (blinking), line-through (strikethrough) None (standard)

CSS 1 also introduces common types of fonts

- serif - classical font (eg Times New Roman)
- sans -serif - sans serif (eg Helvetica or Arial)
- cursive - italic
- fantasy - decorative font
- monospace – Non-proportional font (eg Courier)

Next, CSS 1 defines the URL element so that:

- construction url (), where the source address is specified between brackets Absolute: url (http://www.example.com/ images / logo.png)
- Relative to server: url (/ images / logo.jpg)
- Relative to the current directory: url (images /logo.jpg)

If the URL contains commas, spaces, quotes, or the end of a bracket, these characters can be escape using a backslash.

Then Selectors:

- Type Selector: A - All Type A Elements
- Class selector: A. class - All elements A with attribute class = "class"
- ID selector: A # ID - All elements with ID
- Follower selector: AB - All B elements inside A

The definition of CCS 2 has brought further definitions:

- outline - outer borders
- max-height , max-width , min- height , min- width - Minimum and maximum width or height of the element
- content - adjustable element content
- counter - Chapter numbering tool
- quotes - Quote style
- clip - trimming
- cursor - cursor over the element
- position - possibility to position an element in a row, in a block, absolute, relative, ...
- top, bottom , right , left Marginal values for position: absolute ;
- overflow - Overflow view
- visibility - visibility of the elements
- z-index - Overlap possibilities
- page-break , orphans , widows - typographic rules for page breaks
- Table-layout, border-collapse, border-spacing, caption-side, empty-cells - New

options for viewing tables
- Direction - writing direction

CSS 2 allows you to use new types of selectors:

- Universal: * - Applies to all elements
- Child Selector: A> B - takes into account only those elements B that are nested to A directly
- A sibling selector: A + B - selects all B elements that have the same parent as A and which follow it directly

It also defines selectors using attributes:

- A [attr] - all elements A that have an attr attribute set
- A [attr = value] - all elements A that have the attribute attr = "value"
- A [attr ~ = value] - all elements A with an attr attribute whose value is a list of words separated by a space and just one of these words is the same as "value"
- A [attr | = value] - all the elements A whose value of the attr attribute begins with the "value" string, then the hyphen and the next string

It also introduces a pseudo-element and pseudo- classes system .

- Pseudoelements
    - First line - : first- line
    - First letter (initials) - : first-letter
    - Before - : before
    - For - : after
- Language pseudo-class
    - : lang
- Parental pseudo-class

- Pseudo-Class Links
    - Unvisited link - : link
    - Visited Link - : visited
    - Focused Link - : focus
    - Mouse Link - : hover
    - Active Link - : active

For everything to work properly, order of definitions need to be followed. Each pseudo-class has another priority.

Finally, the latest version of CSS3 that is connected to HTML5 offers:

- Animation - CSS3 directly supports the animation of elements (their properties), animations have been done through DHTML, e.g. jQuery
- additional options for styling backgrounds for block elements, including cropping backgrounds, drop shadows, or rounded edges
- additional rules for overflowing the contents of block elements
- opacity - the degree of opacity of the elements
- additional support for paged content - bookmarks and text splitting options

- flexible block elements
- destination links - how and where to open
- properties for drag'n'drop
- additional attributes for fonts - retrieving fonts from an external source, size adjustment at low readability, font narrowing / scaling
- properties for the generated content - content shortening with expandability, moving elements further on the page
- grids (not yet implemented)
- new features for marquee
- automatic multi-column layout
- Ruby's new features
- properties for read text
- 2D and 3D transformations
- features that work with navigation
- user-defined properties

4. JavaScript

JavaScript is a programming language used on web pages, written directly into HTML code and belongs to client scripts (performed on client side)

JavaScript is:

- interpreted - does not need to be compiled
- Object - uses browser objects and built-in objects
- browser-dependent - works in most browsers
- case sensitive - depends on the font size in the entry
- syntax similar to C, Java, and so on

FROM the essence of JavaScript it has certain limitations:

- It only works in the browser.
- The user can disable JavaScript
- There are different versions of language and browsers, which leads to frequent errors.
- Cannot access files (except cookies) or any system objects.
- Cannot save any data (except cookies).

Nevertheless, JavaScript is widely used and can often be used as a built-in scripting language for many applications.

It can be found in:

- Most extensions for web browsers
- Some NoSQL a dataset like MongoDB or CouchDB they accept queries written in JavaScript.
- Adobe - Acrobat and Adobe Reader, tools in Adobe Creative Suite (Photoshop, Illustrator, Dreamweaver, and InDesign)
- Office suite of applications OpenOffice allows JavaScript to be used as a scripting language.
- Interactive processing of the Max / MSP music signal
- The Apple Logic Pro X audio workstation digital software allows you to create custom MIDI effect plug-ins using JavaScript.
- ECMAScript (JavaScript) was included in the VRML97 VRML scripting standard.
- Game engine Unity 3D supports a modified version of JavaScript for scripting using Mono.
- DX Studio (3D engine) uses JavaScript implementation SpiderMonkey for games and simulation of logic.
- Maxwell Render provides the ECMA scripting engine for task automation.
- Google Apps Script on Google Tables and Google Sites allows users to create custom formulas, automate recurring tasks and also communicate with other products Google as Gmail.
- SpinetiX products use SpiderMonkey JavaScript for scripting in SVG files.

Javascript can also be used as a scripting engine:

- Active Technology Scripting from Microsoft
- Programming language Java in the 6th version presented the package javax.script
- Tool Qt C ++ includes a module QtScript that interprets **JavaScript** as well as a Java package javax.script.

JavaScript can be written to HTML in several ways similar to CSS.

Tag < script > can be used to write the script directly into the HTML stream.

```
< script >
   alert ('Head Up, Worse!');
</ script >
```

Or attach an external script file.

```
< script src = "externi_skript.js"> </ script >
```

Another option is in-line enrolment

```
<P> <A href ="#" onClick =" alert('Hello');">
Click Me </a> </ p>
```

However, the combined use is most often used. External script defines functions, normal writing (using the <script>) variables are initialized and start functions and inline scripts call functions according to events depending on the user's responses.

JavaScript is used on web sites primarily for Booting a page - distinguishing browsers, entering menus from a file, declaring functions, document.write () , or acting as a user an event such as passing a mouse element, clicking, resizing a window, filling in a form ...

5. JavaScript - continued

JavaScript is an object-oriented programming language. It therefore supports the classic object model.

- object.method () - call method (function), command that does something
- object.property - refers to the property of a given object, it has value but does nothing
- object.subobject - reference to a nested object

JavaScript has access

- To Browser Object Objects (Window Class)
- To the elements of the page
- To Math and Date objects , string
- To created objects

Class Window (Window object) is the pinnacle of the hierarchy of objects (classes). Its subdivisions are:

- location - the address of the loaded document
- history - Browsing history
- navigator - Information about the version and browser type
- screen - screen properties (width, height, color)
- frames - work with frames (frame , frameset)
- event - mouse events, keyboard
- document - pictures, forms, links, colors, individual HTML elements ...

The most widely used class is the document class that allows you to manipulate HTML (and other type) document.

Its most important methods are:

- document.images
- document.forms
- document.applets
- document.links
- document.anchors
- document.all
- document.frames
- document.styleSheets
- document.scripts
- document.selection
- document.getElementById()
- document.getElementsByTagName()

The Math class is used to use higher math.

Contains the following methods:

- abs (x)

- exp (x)
- log (x)
- max (x, y)
- min (x, y)
- pow (a, x)
- random ()
- sqrt (x)
- ceil (x)
- floor (x)
- round (x)
- acos (x)
- asin (x)
- atan (x)
- atan2 (x, y)
- cos (x)
- sin (x)
- tan (x)
- Math.E
- Math.LN10
- Math.LN2
- Math.LOG10E
- Math.LOG2E
- Math.PI
- Math.SQRT2
- Math.SQRT1_2 - square root of 1/2

The Date class is used to work with date and time (creating different countdowns, calendars ...)

Main Methods of the Date Class :

- get / setFullYear ()
- get / setMonth ()
- get / setDate ()
- get / setDay ()
- get / setHours ()
- get / setMinutes ()
- get / setSeconds ()
- get / setMilliseconds ()
- get / setTime ()

The String class works with text strings. You can apply a length property to text strings that returns the string length.

Main methods for working with the strings are:

- toUpperCase ()
- toLowerCase ()

- toString ()
- charAt (n)
- charCodeAt (n)
- substring (a, b)
- substr (a, b)
- concat(string1, string2, stringN)
- fromCharCode(code1,code2,..., codeN)
- indexOf (substring)
- lastIndexOf (substring)
- split (separator)

6. XML and Json

XML or eXtensible Mark up Language is a markup language with tandardizovaný W3C. XML can be described as the Standards of format for the exchange of information with international support andinformation m m High High content em. It can easily be converted to other formats, there is an automatic document structure check and supports hypertext and links.

XML efficiency is strongly dependent on the structure, with a poorly designed structure, the XML document is unreadable and inefficient.

Each XML document

- Must have exactly one root (root) element.
- Non-empty elements must be bounded by the start and stop marks. Empty elements may be marked as "empty element".
- All attribute values must be enclosed in quotation marks - single (') or double ('). The opposite pair of quotation marks can be used inside the values.
- Elements can be nested but can not overlap; that is, each (non-root) element must be completely contained in another element.

XML example

<? xml version = "1.0" encoding = "UTF-8"?> XML declaration

< directory > Root element

< person > Nested element 1

< name > Adam < / name > Nested Element 1.1

< phone > 777 777 777 < / phone > Nested element 1.2

< email > adam@adam.com < / email > Nested element 1.3

< / person > Exiting the nested element 1

< person > Nested element 2

< name > Bara < / name > Nested element 2.1

< phone > 666 999 666 < / phone > Nested element 2.2

< email > bara@bara.com < / email > Nested Element 2.3

< / person > Exiting nested element 2

< / directory > Exiting the root element

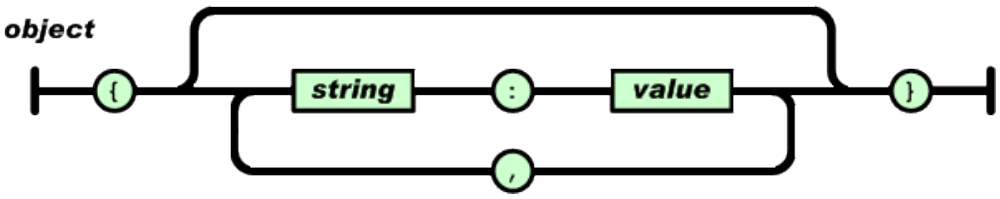
Json - JavaScript Object Notation - is a lightweight format for data exchange, simply readable and writable by a person, written in text format and completely independent of language.

It is composed of two structures:

- Pair Collection - Name / Value
  - Realization: Object , Record , Struct , Dictionary , Hash table, Keyed List, Asociative array
- Assigned list of values
  - Realization: array , vector , list, sequence

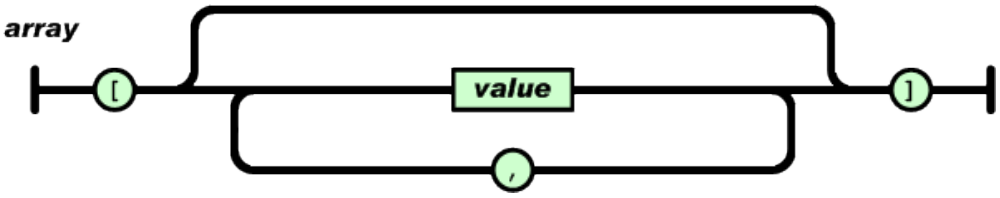
Object - Unordered Set of Pairs Name / Value

Write: {name1: value1, name2: value2}

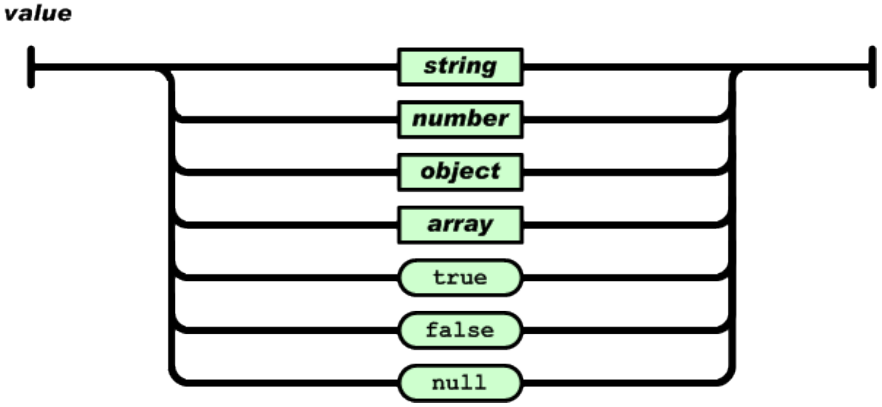


Array - Ordered Value Collection

Writing: [value1, value2, value3]



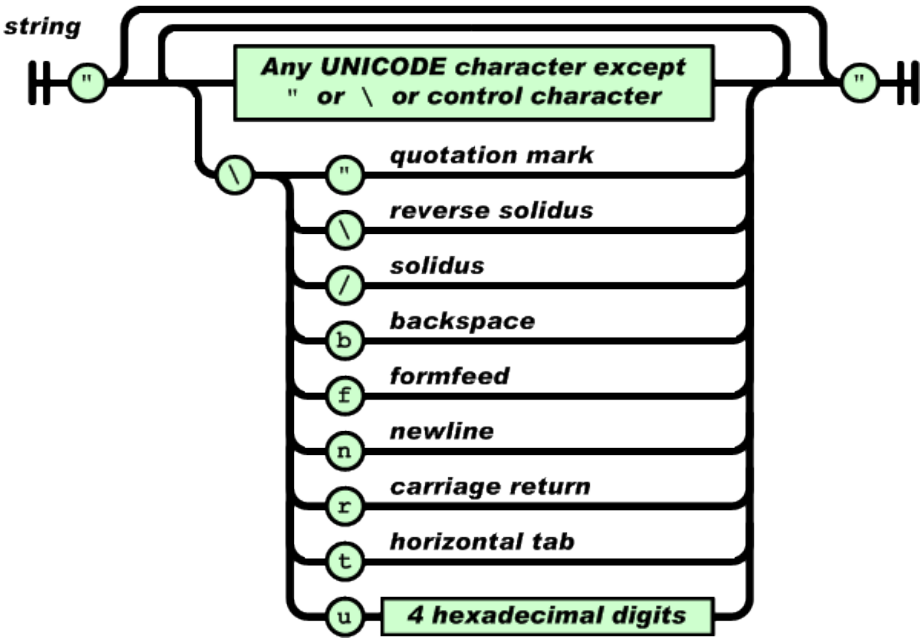
Value - a string enclosed in double quotation marks, number, true , false , null , object, or array. These structures can be nested



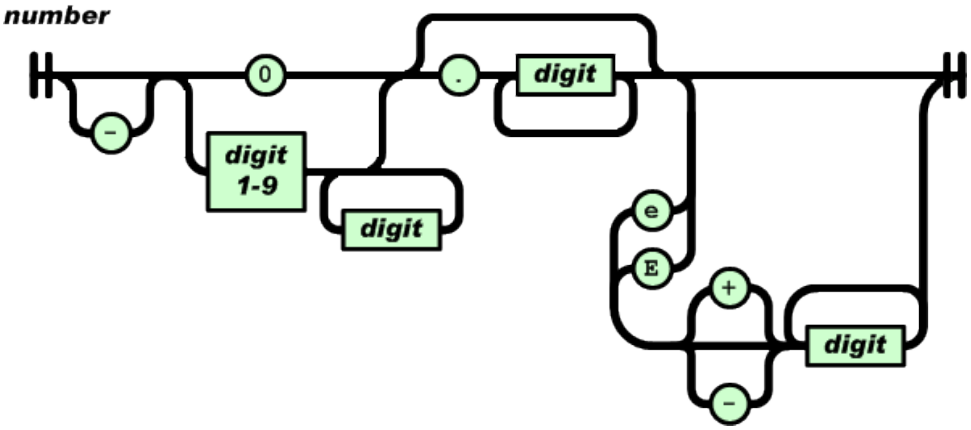
String - zero or more Unicode characters, enclosed in double quotes and using escape

sequence using a backslash.

Character - String with a single character



A number - similar to C and Java numbers - does not use octal or hexadecimal entries



7. Server parts of web technologies

First you need to say what a web server is.

As web server you can designate a computer that is responsible for handling HTTP (S) requests from clients (most often web browsers). By fulfilling the requirements is meant sending a specified URL destination (typically a web page, but also a static text, image, or other file). Web pages are usually HTML documents. Or a computer program that performs the activities described above (a daemon).

HTTP is an Internet protocol designed to exchange hypertext documents in HTML format , uses the so-called Uniform Resource Locator (URL), which specifies the unambiguous location of a source in Internet . HTTP does not allow encryption or data integrity security, HTTPS serves it.

HTTPS is a protocol that enables secure communication on a computer network , uses HTTP along with SSL or TLS .

HTTPS benefits are

- authentication
- confidentiality of transmitted data
- integrity of content
- the ability to use the HTTP/2 protocol
- Google search favours

Disadvantages:

- decrease in performance on older hardware
- the need for a certificate and its renewal
- does not allow blocking specific URLs only to block the entire site
- slightly more complicated web server configuration
- possible complications with older web browsers

Each web server is connected to a computer network and receives HTTP requests. Requirements handle and the computer returns a response, usually an HTML document. (or text, image , etc.) . The server response is in the HTTP format, with a header containing the status code followed by the content itself.

Web server replies (status code)

- 2xx - successfully completing the request
- 3xx - redirect problems
- 4xx - errors related to request handling (page not available, etc.)
- 5xx - internal server errors

Source of Server information can be:

- Static content - pre-prepared data files (HTML pages), significantly faster than dynamic.
- Dynamic Content - Based on the request, data is collected (read from a file, database, or some end device), formatted and ready for presentation in HTML format and provided to

a web browser

- Dynamic Content Creation - A variety of technologies (Perl, PHP, ASP, ASP.NET , JSP, Python, etc.) can provide much more information and can respond to various "ad hoc" queries

In practice, both approaches are combined - caching, node.js, ...

In real-time traffic, the Web server may be overloaded.

Overload symptoms:

- slow server response (from units to hundreds)
- Errors 500, 502, 503, 504
- The TCP connection is forced to restart before the answer arrives
- the server sends incomplete content (this behaviour is mostly caused by an error)

Reasons for overload:

- Classic overload (too many people join at the same time but not for attack)
- DDoS Attack, A computer virus that attacks many computers and forces them to connect
- Internet boot
- Overloading physical network
- Content is spread over multiple servers and none of them available. All queries must be served by only one server

Techniques to avoid overload:

- network traffic control using firewalls, HTTP traffic managers and traffic shaping
- using web caches
- using different domain names for static and dynamic queries
- use of different domain names and / or computers to separate large files so that the small ones can be stored in cache
- using multiple web servers on one computer, each with a custom network card
- using multiple computers connected together and outward looking like one big server
- adding more hardware (RAM, CPU)
- tuning the used software

8. Server parts of web technologies II

There are several types of web servers. The most common are:

- Apache HTTP server
- IIS - Internet Information Service
- nginx
- GWS - Google Web Server

Apache HTTP server is open source software web server for GNU / Linux, BSD, Solaris, Mac OS X, Microsoft Windows and other platforms.

It supports a wide range of functions - compiled modules expanding core programming languages on the server side (Perl, Python, Tcl, PHP ...), various authentication schemes (mod_access, mod_auth, mod_digest and mod_auth_digest), support for SSL, TLS (mod_ssl), a proxy (mod_proxy), URL rewriter known as rewrite engine from mod_rewrite, configuration of log files (mod_log_config) and filtering (mod_include and mod_ext_filter)

Includes an external data compression module (mod_gzip), an open source module for protecting and preventing web applications from attack (mod_security)

Logs can be analyzed using browser and scripts such as AWStats / W3Perl or Visitors.

Support for many graphical environments (GUIs)

Virtual hosting - One Apache installation on one physical computer serves multiple web-sitesIIS - Internet Information Service is a software web server with a collection of ex-pansion modules, created by Microsoft for the Windows operating system .

It supports a number of protocols - HTTP, HTTPS, FTP, FTPS, SMTP and NNTP

Modules for IIS 7.5

- FTP Publishing Service - Publish content securely to IIS 7 servers with SSL authentication and data transfer.
- Administration Pack - Support for managing UI management features in IIS 7, including ASP.NET privileges, custom errors, FastCGI configuration, and request filtering.
- Application Request Routing - Provides a proxy- routing module that passes HTTP requests to content servers based on HTTP header of server variables and alignment algorithms.
- Database Manager - Easily manage local and remote databases within IIS Manager.
- Media Services - Connects the media platform with IIS to manage and manage multimedia and other web content.
- URL Rewrite Module - Provides a rewrite mechanism that changes the URL request before it is processed on a web server.
- WebDAV - Allows site authors to publish content securely on IIS 7 servers.
- Web Deployment Tool - Synchronizes IIS 6.0 and IIS 7 servers. Changes IIS 6.0 to IIS 7

Nginx is a software web server with load Management and Reverse Proxy with Open Source Code. It works with HTTP (and HTTPS), SMTP, POP3, IMAP, and SSL protocols. It

focuses primarily on high performance and low memory demands. It is extensible on Unix, Linux and Unix- like systems under BSD, there are variants for Solaris, macOS and MS Windows .

The basic objective of m em Nginx is the fast distribution of static content, the possibility of distributing the load on other servers according to the set priority.

The system allows you to define a backup server to which the Nginx request passes, unless the primary server responds to a specified limit

Inbound requests Nginx processes and processes asynchronously

An incoming HTTP (or HTTPS) request first attempts to search in its cache (it has a configurable size and retention time), if it finds it, it answers straight away . Otherwise, they turn to one of the defined set of servers (each server has a defined priority). If the server is able to answer within a defined time, he / she will reply; otherwise, it turns to the backup server (of course, if defined). The answer, if it can, store in its cache, and subsequent queries to timeout lifetime cache handles being cached.

Possibility to set connection limit from one IP address

Nginx is a modular system

One of the modules - GEO Locations - allows, for example, country to pass requests for defined servers, or to disable access to sites from some countries

Module redirects according to defined rules, password security, gzip compression support, streaming (FLV, MP4)...

9. PHP: Hypertext Pre-processor

Programming paradigm of PHP - imperative, object-oriented, procedural, reflective

It was created in 1995 , and his author is Rasmus Lerdorf . The first release was June 8, 1995, and the latest version is 7.2.0 (November 30, 2017)

PHP is characterized by weak and dynamic type control.

The main implementations of PHP are Zend Engine, Phalanger , Quercus , Project Zero , HipHop

PHP is a programming language designed specifically for programming dynamic web sites and web applications. The scripts are done on the server side - the user is transferred to the result of their activity. PHP script interpreters can be called using the command line, HTTP queries or web services. The syntax of language is inspired by several programming languages (Perl, C, Pascal, and Java)

PHP language is platform-independent, the differences in different operating systems are limited to several system-dependent functions, and scripts can usually be transferred between operating systems without any modifications. It supports many libraries for various purposes - eg. Word processing, graphics, file handling, access to most database systems (eg. MySQL, ODBC, Oracle, PostgreSQL, MSSQL), ranks among Internet protocols (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP ...).

PHP is the most widely used scripting language for the web

PHP properties:

- PHP language is dynamically typed - the data type of the variable is bound to a value, not a variable.
- Fields are associative
- Strings can be written in PHP in 2 different ways:
    - to quote the quotes (the evaluation is done by replacing the variables inside)
    - Closing in apostrophes (only the escape sequence \ ' is replaced).
- Variables can be created and disturbed
- Constants can be defined, cannot be deleted
- Variables have their visibility levels and rules for their persistence.
- Supports references that can be used to store references to any other variables or variables in the field
- As a reference, the function parameters can also be called - for each variable, it records how much it is referenced by the reference, and therefore decides when it can cancel the variable.

Advantages:

- PHP is dedicated to websites.
- Extensive set of functions in the PHP basic library (over five and a half thousand),

additional features in PECL .

- Native support for many database systems.
- Multiplatform (especially Linux and Microsoft Windows)
- The ability to use native operating system features (possibly incompatibility with another OS)
- Strict learning curve.
- Huge support for hosting services
- A huge number of projects and codes that can be used for free (WordPress , phpBB, and more).
- Relatively decent documentation
- Very free license

Disadvantages:

- PHP language has long been defined only by its implementation, the official language specification was announced at the end of July 2014
- Inconsistent naming of features
    - strpos (), strchr (), but str_replace (), str_pad ().
- Non-uniform nomenclature of function groups
    - mysql_XXXX , imap_XXXX , json_XXXX (with underscore) versus imageXXXX , bcXXXX , gzXXXX (without underscore).
- Non-uniform order of parameters, eg: array_map () vs. array_filter ().
- Although the language supports exceptions, its library is rarely used.
- Weaker Unicode support, only through the PHP library (in versions after PHP 5, the Unicode string should be the basic type).
- In the standard distribution of missing debugging (debugging) tool.
- After processing the request, it does not maintain the context of the application, it always creates it again (weakens performance).

PHP language is not only for small projects and pages; it can be programmed into any application.

Selected major projects in PHP:

- MediaWiki - software for creating wiki web projects,
- phpBB - a package for running a web forum
- WordPress - a publishing system for blogging and similar applications
- Adminer - Web application for managing MySQL database system
- phpMyAdmin - Web application for managing MySQL database system
- Texy! - Intuitive syntax compiler for formatting text to HTML
- Nette Framework - a framework for creating web applications in PHP
- Facebook

10. PHP II - Syntax

PHP script can be tagged in HTML in several ways

<? [PHP code] ?>

<? php [PHP code] ?>

<SCRIPT LANGUAGE = " php "> php [PHP code] </ SCRIPT>

The individual instructions (commands) are separated by a semicolon. Comments are preceded by a double slash or grid. Multi-line comments are written between the forward slash - star slash (// , # , / * multiline text * /)

Types of variables in PHP

- Variables
- Logical type - Boolean - True / False, written as TRUE and FALSE (size does not matter)
- the integer type - Integer - a positive or negative number (a zero) from some -2 to + 2 trillion trillion
- Decimal number - Float , Real - with accuracy to 14 decimal places
- String - String - text strings

The type of variable is determined at the time the value is assigned, during the program the variable can change its type, either through code instruction or as a result of a calculation . Each variable must have a unique name, starting with the dollar sign ($) and without a space followed by a name. The first character of that naming must be either the letter az or the underscore. It cannot be a number or anything else. Variable names distinguish between uppercase and lowercase letters. For assignment, a character equal to (=) is used

PHP allows you to define three types of fields:

- Indexed
- Associative
- Multi-dimensional

These are used as lists, dictionaries simulation and collection of elements. We can work with arrays as stacks or queues, and they can also represent tree structures (a field element may be a field) .

Fields can be returned from PHP (database) functions.

PHP is among the PPE languages.

```
Class ClassName

{
```

```
var $ VariableName

function Function Name (parameters)

{

        body function
```

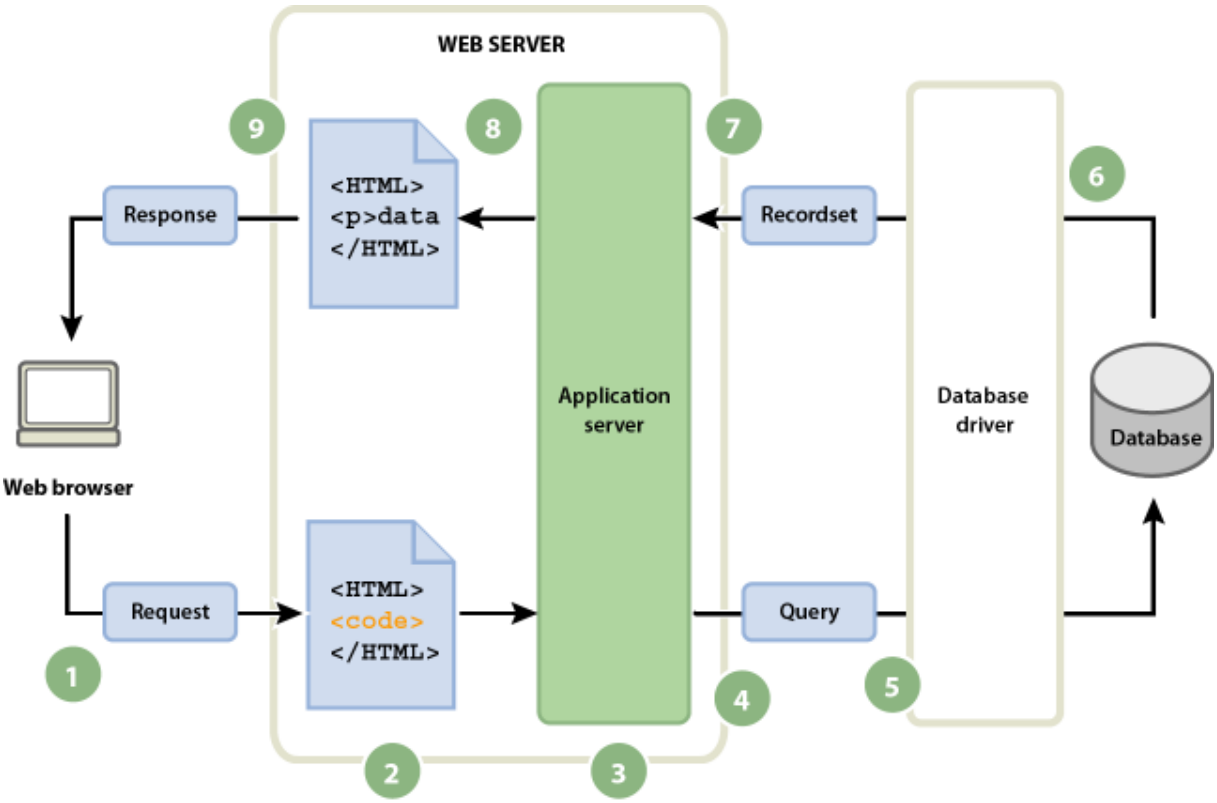
Beware of inheritance - PHP does not have private methods (functions).

In addition, PHP allows you to enter and change NONDECLARED attributes!

Many finished libraries and parts of the PHP object code are used, you can only have one "level" of objects without inheritance. An object can exist without methods, with attributes only. But sometimes it is better to use the field. Someone in PHP, for "assigning" each attribute, creates a method or methods. When using objects, it's more than ever that you should think and follow conventions (somebody starts with methods that should not be called by an underscore).

11. Web applications working with databases

How to process a web request into a database?



1. Request - request
2. HTML code
3. Application server
4. Query - Query
5. Database driver
6. Output from database
7. Recordset for the server
8. Translate to HTML
9. Answer - Response

The basis for working with the database is SQL.

SQL - Structured Query Language - with a tricky language for working with relational databases

The database is a file system with a fixed record structure, where the files are interconnected using the keys.

Database types

- Hierarchical database
- Network database
- Relational databases
- Object database
- Object-based relational databases

Database Objects

- TABLE - a basic database object used to directly store data in the relational database storage space
- VIEW - a database object that provides the user with a preview of the data contained in the table
- INDEX (KEY) - for speeding up search and query processes, defining a unique value with table peeling, search optimization
- CONSTRAINT - allows you to create restrictions on the conditions that must be met for the values of its columns when inserting or changing records
- TRANSACTION - a group of commands that convert the database from one consistent state to another
- TRIGGER - defines the actions to be performed in the event of a defined event above the database table

SQL Data Handling Commands (DML)

- SELECT - selects data from the database, allows selection of sub-conditions and data sorting.
- INSERT - inserts new data into the database.
- UPDATE - Changes database data (editing).
- MERGE - INSERT and UPDATE combine the data (if there is no corresponding key) if it exists, then modify them in the UPDATE style.
- DELETE - removes data from the database.
- EXPLAIN - a special command that displays the SQL statement processing procedure. Helps the user to optimize commands so they are faster.
- SHOW - a less common command allowing you to view databases, tables or their definitions

SQL commands for data definition language (DDL)

- CREATE - Creating new objects.
- ALTER - Changes to existing objects.
- DROP - removing objects.

SQL commands for data management (DCL)

- GRANT - a command for assigning permissions to the user to certain objects.
- REVOKE - Removal command for user.
- START TRANSACTION - starts the transaction.
- COMMIT - Transaction Confirmation.

- ROLLBACK - Cancels the transaction, returns to the original state.

Keywords SQL for querying

- TOP - returns the first N rows
- LIMIT - Limit the number of lines returned by the SE LECT command
- JOIN (FULL LEFT RIGHT INNER CROSS) ON - combining the result of a SELECT query from two input sets (typically a table to a relational database)
- UNION - Unification of query result from two or more SELECT query input sets
- ORDER BY - sort the entries selected using the SELECT statement
- WHERE - limits the selection of rows from tables using conditions
- GROUP BY - the aggregation of records selected by the SELECT statement
- WITH ROLLUP - with a standard dump, a line with a NULL value appears instead of a column, according to which the data is aggregated (if specified)
- HAVING - allows you to restrict the rows that are processed by the aggregation function

Databases and database servers

- Microsoft Access
- MySQL
- Oracle
- Microsoft SQL Server
- SQLite

12. DOM - Document Object Model

DOM is the Document Object Model, API (application programming interface) that defines a common standard for accessing any valid HTML document or a properly structured XML document is entirely independent of the programming language

Using DOM, it is possible to handle individual elements (objects) using JavaScript.

DOM definitions describe individual levels

- Level 0
    - Intermediate DOM support that existed prior to creating DOM Level 1. For example, DHTML Object Model developed by Microsoft, or Netscape's unnamed Intermediate DOM . Level 0 is not a formal specification published by W3C, but is used as a comprehensible abbreviation referring to things existing before the standardization process.
- Level 1
    - Navigate in the DOM (HTML and XML) of the document (or its tree structure) and manipulate the content (including adding elements). Specific HTML elements are also included.
- Level 2
    - Support for namespaces, events, and filtered views.
- Level 3
    - Standardized loading and storage mechanism and support for XML schemas. Allows dynamic insertion of content into a document and adds new methods and properties.
- Level 4
    - Merging the previous standards DOM Level 3 Core, Element Traversal , Selectors API Level 2 DOM Level 3 Events and DOM Level 2 Traversal and Range and simplification approach and existing standards, particularly specifications JavaScript and HTML5. Specifications will also simplify frequent DOM operations.

Basic ideas of DOM are to consider all HTML elements as objects. Then each object has properties - attributes - and can respond to events.

Each object must be identified; it is necessarily for changing the properties or content by using a script. Identification is by id or name.

DOM - Universal Properties and Methods for Crawling and Reading (Usage: Document.Name)

- documentElement - returns the root (root) document element
- getElementsByTagName () - returns the field of all elements of the given name
- parentNode - returns the parent object of the object
- nextSibling - returns the next sibling of the current object if it exists, otherwise it returns null
- previousSibling - returns the previous sibling of the current object if it exists
- firstChild - Returns the child's first child
- lastChild - returns the child's last child
- childNodes [] - returns the array of all objects (nodelists) that are the children of the object
- nodeName - returns the name of the object
- nodeValue - Returns the value (content) of the object
- data - returns the value (content) of the Text object
- className - returns the value of the class attribute (HTML only)
- id - returns the value of the attribute of the object id, which must be an element type, for HTML only
- title - returns the value of the title attribute of the element, HTML only
- item () - specifies a specified object field (nodelist) specified by the index

DOM - Methods for manipulating nodes

- createElement (name) - create a new element
- setAttribute (name, value) - setting the attribute
- createTextNode (value) - Creates a text node
- splitText (division) - splitting the text node into nodes two
- normalize () - Merge sister nodes of the Text type into one node
- appendChild (object) - Adds a child node
- insertBefore (object, object) - 2 parameters - the first is the node to be inserted and the other one is the node to which we will insert
- cloneNode (true | false) - creates a copy of the object
- replaceChild (object, object) - 2 parameters - the first parameter is a node that replaces the node given as the second parameter
- removeChild (Object) - Applies to the parent of the node to be removed
- removeAttribute (name) - Removes the attribute node

DOM can also be used to efficiently create dynamic tables or dynamically modify page formatting using CSS.

Interreg

EVROPSKÁ UNIE

Rakousko-Česká republika

Evropský fond pro regionální rozvoj

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

VŠTE

EUROPEAN UNION