1. Struktura vědecké práce

Studie může být:

- odborná písemná práce, obvykle kratší vědecký článek, odborné pojednání
- sborník studií (literárněvědných, o životním prostředí apod.)

Vědecké zkoumání = činnost, která je založena především na zjišťování a třídění faktů, získávání dat, ověřování experimentů, a to vše při snaze o logičnost a systematičnost

Hlavním úkolem vědecké a odborné práce je získávání nových poznatků, znalostí a vědomostí za využití systematického a kritického myšlení.

Ať už se autor rozhodne jakkoliv (v jakékoliv formě bude odbornou práci prezentovat), musí vždy dodržet obecně platná pravidla o jazykové normě, struktuře a formální úpravě.

Podle Umberta Ecca má práce **vědecký charakter**, pokud splňuje:

1. Předmětem výzkumu musí být nějaký identifikovatelný předmět, který musí být definován tak, aby byl poznatelný a identifikovatelný i pro ostatní.
2. Výstup výzkumu musí být takový, aby námi zvolený předmět zkoumání sdělil nové věci (věci, které nebyly vyřčeny)
3. Výzkum musí být užitečný a prospěšný pro ostatní
4. Výzkumná práce musí poskytnout předpoklady pro potvrzení nebo vyvrácení

Druhy a žánry odborného textu

Vědecký či odborný text = stylistický útvar, jehož cílem je zaujímat stanoviska, vysvětlovat myšlenky, informovat, komunikovat o teoriích různých vědních oborů s odbornou veřejností.

Autor již k známým publikovaným faktům přidává vlastní pohled a poznatky vědeckého charakteru získané vlastním výzkumem nebo odvozené z dřívějších prací. Celkově by měl daný odborný text směřovat k úplnosti a vnitřní uspořádanosti předávané informace.

Druhy

Odborné texty se tak dělí **na 4 druhy**:

- **Vědecké** – texty určené odborníkům v daném oboru, autoři zpracovávají teorii jednotlivých vědních oborů a usilují o jednoznačnost vyjadřování
- **Populárně-naučné** – texty, které zpřístupňují výsledky vědy široké veřejnosti (ta se o problematiku zajímá ale ne na profesionální úrovni)
- **Učební** – didaktické texty – učebnice, skripta
- **Esejistické** – texty na pomezí odborného, publicistického a někdy uměleckého stylu

Z uvedených druhů pak vyplývají **funkce odborného stylu**:

- **Odborně sdělná** – základní funkce všech odborných textů, autor musí zachovávat objektivní přístup ke sdělované skutečnosti (bez emocí a předsudků)
- **Poznávací, systematická, vědecká** – souvisí s funkcí odborně sdělnou, tvůrce i příjemce textu jsou vědci (odborníci)
- **Vzdělávací a řídicí** – text vede čtenáře tak, aby porozuměl problému a naučil se něco. (učebnice, skripta)
- **Odborně kritická** – zhodnocení textu (odborný posudek, recenze)
- **Kvalifikační** – jedná se o závěrečné práce, které vedou k získání titulu

    - Bc.; Mgr. a Ing.; PhDr.; JUDr.; RNDr.; doc.

Odborné texty se ještě dělí **podle původu obsahu** na:

- Práce původní – zdroj nových informací a závěrů
- Práce přehledové – rekapitulují současné dostupné informace
- Práce komparativní – srovnávají současně dostupné informace

Žánry

Vedle druhů odborných textů rozlišujeme i žánry

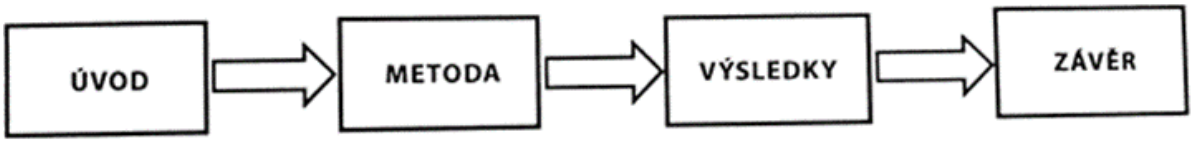
Žánr = zvláštními specifickými znaky odlišený typ kompozičního uspořádání vědecké práce

Typy žánrů:

- **Abstrakt** – nejkratší odborný text, shrnutí celé knihy či práce, zhruba 150 – 250 slov, vždy v českém a anglickém jazyce
- **Resumé** – delší shrnutí práce
- **Recenze** – zhodnocení vědecké práce, měla by být subjektivní, ne delší jak 7 stran
- **Rešerše** – soupis subjektivně nejdůležitějších bodů a myšlenek odborného textu, slouží k rychlému pochopení a orientaci v tématu
- **Konspekt** – písemný záznam obsahu a podstatných myšlenek odborného díla, je to prostě výpisek z odborné práce, často obsahuje i komentář autora
- **Anotace** – na několika řádcích autor vysvětlí, co se za názvem tématu skrývá.
- **Teze** – zhuštěný výklad hlavních (základních) myšlenek teprve připravované publikace. Cca 10 stran
- **Odborný esej** – slohově vybroušená úvaha, má za cíl navrhnout řešení nějaké odborné otázky. Poměrně svobodný jazykový útvar, který klade vysoké nároky na odbornost
- **Polemika** – vědecký spor, jedná se o vyargumentované hájení vlastního názoru
- **Referát** – zpráva o odborném problému a jejich výsledcích
- **Koreferát** – je zaměřen pouze na jeden aspekt problému z přechozího referátu
- **Vědecká stať** – odborný žánr, ve kterém autor koncipuje vlastní, nové řešení problému či tématu
    - Teoretická – výsledek náročné intelektuální práce, autor předkládá vždy nové řešení daného tématu
    - Empirická – shrnutím výzkumného problému a jeho vyvrcholením, odpovídá na hypotézy, které buď potvrdí, nebo vyvrátí
    - Přehledová – přehled o jednom problému nebo o jedné problematice ze všech úhlů pohledu
- **Odborný a vědecký článek** – jedná se o kratší stať, které pojednává o nějakém problému
- **Příspěvek do sborníku** – určen k tomu, že hlavní myšlenky a závěry budou autorem předneseny na vědecké konferenci
- **Monografie** – vědecká kniha, ve které se autor (odborník) věnuje ohraničenému tématu
- **Učebnice** – odborné texty určené ke vzdělávání čtenářů
- **Seminární**, popř. **ročníková práce** – zadává se studentům středních i vysokých škol, cílem je aby studen získal zkušenosti a dovednosti
- **Závěrečná písemná práce**
    - Bakalářská
    - Diplomová
    - Rigorózní – po tom co uděláš magistra, můžeš složit rigorózní zkoušku
- **Disertační práce**

Uspořádání odborné práce (pr. Norma pro strukturu odborné práce – IMRaD (Introduction, Methods, Results, Discussion)

Obrázek 6 | Schéma odborné práce



Zdroj: IMRaD

Principy:

Téma a název práce

Na začátku každé práce je myšlenka, kterou chceme zpracovat – téma práce. Téma práce musí korespondovat s cílem a přínosem práce. Mělo by odpovídat schopnostem a zájmům autora. Téma musí být konkrétní a nesmí sklouznout k obecnostem. Název práce pak musí odpovídat jak obsahu, tak i zvolenému tématu.

Cíl práce

Měl by být definován v úvodu práce. Vhodné je zasadit cíl do širších souvislostí a stanovit tak, co s prací souvisí a co ne. Potřeba též odůvodnit, proč jsme stanovili daný cíl. Cíl musí jasně a přesně charakterizovat předmět řešení. V závěru pak musí autor napsat jeho dosažení. Po stanovení cíle je možné přejít ke zpracování osnovy textu

Osnova hlavního textu

Hlavní text práce je vždy ohraničen úvodem a závěrem. Autor by měl stanovit osnovu práce před tím, než se pustí do psaní. Čím lépe je promyšlené schéma osnovy, tím lépe se práce bude psát. Zásady pro osnovu:

- **Přehlednost** – jednotlivé části by měly být sdruženy ve skupinách, aby se dalo práci snadněji členit
- **Jasnost** – osnova musí být dostatečně podrobná a pochopitelná
- **Shoda s tématem** – obsah práce se nesmí odchýlit od tématu
- **Proporcionalita** – každá kapitola i podkapitoly musí být úměrné svému významu
- **Souvislost** – mezi jednotlivými kapitolami musí být návaznost a logická souvislost
- **Ucelenost** – všechny části jsou nezbytné k vytvoření vyváženého díla
- **Úplnost** – téma by mělo být zpracováno komplexně, vyčerpávajícím způsobem

Současný stav sledované problematiky

Autor by měl uvést dostupné poznatky, které jsou spojené s tématem. Jedná se o autorovo východisko, rešerši

Výsledky a interpretace

Nejvýznamnější část práce. Výsledky autor musí logicky uspořádat a poznatky dostatečně zhodnotit. Měl by odpovědět na otázky položené v úvodu práce, vysvětlit výsledky a jak k nim dospěl.

Závěr a doporučení

Obsahuje stručné výsledky celé práce v souladu se stanoveným cílem. Měla by sumarizovat celou práci.

Základní členění práce

Společné prvky všech vědeckých prací:

- **Název** – musí vystihovat práci jako celek, krátký, jednoduchý a výstižný
- **Autor** – uvádí se bez titulů, ten se zaznamenává až na konci textu
- **Abstrakt** – souhrn odborného textu, kolem 150 – 250 slov max., měl by zahrnovat cíle, metody, zpracování

- **Klíčová slova** – termíny, kterými autor charakterizuje obsah odborné práce
- **Obsah**
- **Úvod** – uvedení do tématu či problematiky
- **Text** – člení se do kapitol a podkapitol, začátek nové kapitoly na nové stránce
- **Závěr** – nejdůležitější část práce. Odpovídá, zda byl cíl splněn nebo ne.
- **Seznam zdrojů** – nachází se na konci, souhrn údajů o citované publikaci
- **Přílohy** – vkládají se až za seznam použité literatury, doplňující tabulky, grafy. Při větším počtu příloh je nutné uvést jejich seznam

2. Bibliografická data a reference

V rámci přípravy vědecké práce se musíme zabývat otázkou dostupnosti pramenů pro naše zkoumání. O tom, zda přijmeme téma, spolurozhoduje i skutečnost, jsou-li zdroje informací dostupné. K tomu je vždy nutné ověřit:

- kde se příslušné informace nacházejí (knihovna, internet, vědecké časopisy, přednášky…)
- přístupnost zdrojů (archivy – čas dodání, prezenční studium materiálů, informační agentury, média…)
- naši schopnost pracovat se zdroji (jazyk, písmo…)

Prameny vědeckých informací

Dvě základní skupiny:

- primární prameny
- sekundární prameny

Při hledání informací je vhodné začít orientací v sekundárních pramenech. **Sekundární prameny** samy o sobě nepřinášejí nové odborné informace, ale ukrývají v sobě odpověď, kde se nacházejí primární informace. Také slouží k rychlé orientaci v základních pojmech nebo k vyhledávání informací o daném problému (slovníky, lexikony, encyklopedie…)

Primární prameny jsou důležitými zdroji informací ke zpracování daného tématu (monografie, učební texty, odborné časopisy, právní literatura…; vše v písemné i elektronické podobě)

Prvním zdrojem informací bývá internet. Informace z internetu je však nutné si pečlivě ověřovat. Také pomocí následujících otázek:

- Je u textu uveden autor, nebo je možné autora textu zjistit?
- Známe jméno autora, nebo lze autora přiřadit k nějaké konkrétní instituci či organizaci?
- Jedná se o seriózní nabídku například vysoké školy, úřadu, vědecké instituce…?
- Jak aktuální je internetová stránka?
- Je interpretace tématu kompetentní a ověřitelná?
- Není téma zpracováno příliš subjektivně a jednostranně, jsou uvedena i jiná mínění?
- Jsou součástí textu údaje o zdrojích a dalších informacích? Lze je ověřit?

Bibliografické citace

Pro citace existují jak pragmatické, tak právní důvody. K právním patří **Zákon č. 121/2007 Sb. o právu autorském**, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Pragmatickým důvodem je možnost ověření uvedených tezí, uvedení tvrzení autora do širšího kontextu a dokázání tak vlastní znalosti tématu, informační a autorská etika….

Způsob citování a bibliografické odkazy určují **citační norma APA.** Citační styl APA 7. ed. je preferovaným citačním stylem pro celou VŠE. Styl APA vychází z Publikační příručky Americké psychologické asociace (American Psychological Association, 2019). Prvky citace se dělí na povinné a nepovinné. **Povinné prvky** jsou autor, název díla, místo a

nakladatelství, rok vydání, ISBN (monografická publikace) nebo ISSN (seriálová publikace). **Nepovinné prvky** jsou jména překladatelů, pořadí vydání, údaje o rozsahu díla, název edice, náklad…

Pojmy:

Bibliografický záznam je souhrn údajů o citované publikaci nebo její části umožňující její identifikaci.

Bibliografická citace je převzetí textu jiného autora do naší odborné publikace. Může se jednat o doslovnou citaci nebo parafrázovanou citaci.

Odkaz na citaci je odvolání se v textu na citaci uvedenou na jiném místě. Odkaz je identifikace dokumentu, ze kterého autor cituje.

Bibliografie je uspořádaný seznam záznamů o dokumentech, které se týkají předmětu práce, ale nebyly bezprostředně použity k jejímu napsání.

Citační etika vyžaduje, aby autor zveřejnil veškeré informační prameny, které použil.

Příklady citací - https://knihovna.vse.cz/citace/priklady/

Odkaz na citaci

Odkaz slouží k identifikaci publikace, která byla citována nebo parafrázována. Propojení odkazu s bibliografickou citací lze provést třemi způsoby:

- citování podle prvního údaje a roku vydání (tzv. Harvardský styl), vše je uvedeno v závorce bezprostředně za citací. Příklad: (Toman, 2011).
- citování pomocí průběžných poznámek, zdroje jsou uvedeny ve zkrácené verzi za číslem pod čarou na konci stránky. Příklad: 8) Toman, P. Informatika. Praha, 2011, s. 23.
- citování pomocí číselných odkazů – číslice v závorce odkazují na dokumenty v pořadí, jak byly poprvé odkázaný Po sobě následujícím odkazům ke stejnému dokumentu je přiřazeno stejné číslo. Pod tímto pořadovým číslem je uveden zdroj v seznamu literatury. Příklad: citace (31)

Citační manažery

Citační manažery jsou elektronické nástroje pro vytváření, správu, ukládání a sdílení vytvořené či nalezené citace. Existují manažery placené i naplacené.

Zotero

Volně dostupný manažer, který je nadstavbou prohlížeče Firefox. Umožňuje ukládat a spravovat dokumenty z webu i odjinud. Spousta citačních stylů i možnost vlastních. Export do bibliografických citací v různých formátech. Možnost sdílení mezi ostatní uživatele. Od verze 3.0 k dispozici jako samostatná aplikace propojitelná s Google Chrome a Safari.

Citace.com

Jednoduchý nástroj s generátorem citací. Není třeba registrace ani přihlášení. V případě využití jako citační manažer registrace nutná je. Také komerční verze Citace Pro určená pro instituce (lepší možosti a funkce).

ReferenceManager

Umožňuje hledat bibliografické záznamy či plné texty na internetu i ve vědeckých databázích ISI Web of Science, PubMed… Kladem je možnost sdílení databází citací. Vhodný zejména pro knihovny, laboratoře a velké výzkumné ústavy. Plná verze je placená.

3. Výzkum kvantitativní a kvalitativní

výzkum = co dělají vědci s použitím svých vědeckých metod

Kvalitativní výzkum

- cílem kvalitativního výzkumu je vytváření nových hypotéz, nového porozumění, nové teorie
- logika kvalitativního výzkumu je **induktivní***
- kvalitativní výzkum je proces hledání

„Kvalitativní výzkum je proces hledání porozumění založený na různých metodologických tradicích zkoumání daného sociálního nebo lidského problému. Výzkumník vytváří komplexní (holistický) obraz, analyzuje různé typy textů, informuje o názorech účastníků výzkumu a provádí zkoumání v přirozených podmínkách“

- na začátku výzkumného procesu je pozorování, sběr dat
- poté výzkumník pátrá po pravidelnostech existujících v těchto datech, pátrá po významu těchto dat, formuluje předběžné závěry a výstupem mohou být nově formulované hypotézy
- také vyhledává a analyzuje jakékoliv informace, které přispívají k osvětlení výzkumných otázek (ty vznikají v průběhu výzkumu)

Charakteristiky (Hendl, 2005):

- kvalitativní výzkum se provádí pomocí delšího intensivního kontaktu s terénem nebo situací jedince či skupiny jedinců
- výzkumník se snaží získat integrovaný pohled na předmět studie, na jeho kontextovou logiku, na explicitní a implicitní pravidla jeho fungování
- používají se relativně málo standardizované metody získávání dat, které zahrnují přepisy terénních poznámek z pozorování a rozhovorů, fotografie, audio a videozáznamy, deníky apod.
- hlavním úkolem je objasnit, jak se lidé v daném prostředí a situaci dobírají pochopení toho co se děje, proč jednají určitým způsobem a jak organizují své všednodenní aktivity a interakce
- data se **induktivně*** analyzují a interpretují

**Induktivní přístup vychází z konstruktivismu a pracuje především z kvalitativních dat. Sběru dat se využívá k získání několika různých pohledů na daný problém. Na základě zjištěných poznatků se vytváří hypotézy, které se následně testují za účelem zobecnění a tím i vzniku „nové“ teorie. Induktivním přístupem bývají spojovány případové studie pro řešení otázek „proč“ a „jak“ a vyžaduje hlubší pochopení daní problematiky.*

-standardizace (normalizace) v kvalitativním výzkumu je slabá, a proto má poměrně nízkou reliabilitu (volná forma otázek a odpovědí nevynucuje taková omezení jako kvantitativní výzkum) - proto potenciálně může mít vysokou validitu (platnost)

Kvantitativní výzkum

Spočívá v analýze dat, která mohou být získána buď přímým pozorováním nebo dotazováním

- pozorování = technika sběru informací založená na zaměřeném, systematickém a organizovaném sledování aspektů, fenoménů, které jsou předmětem zkoumání (př. zjišťování cen výrobků v prodejnách, měření času potřebného pro vyhledání relevantních webových stránek odpovídajících zadanému dotazu)
- dotazování = uskutečňuje se pomocí nástrojů (záznamový arch, dotazník - strukturovaný (s uzavřenými otázkami) a nestrukturovaný (s otevřenými otázkami) a vhodně zvolené komunikace výzkumníka s nositelem informací, díky nimž řešitel výzkumného projektu získá žádoucí primární údaje

Zjištěná data jsou hodnoty proměnných veličin (**proměnných**)

- kvantitativní proměnná (cena výrobku, zjištěný čas)

- V některých případech je obtížné zjistit kvantitativní údaj přesně (odhad ročních výdajů domácnosti na potraviny), proto dotazovaní respondenti vybírají z nabízených intervalů takový, v nichž by se zjišťovaná hodnota měla nacházet
    - ty se obvykle označují pořadovými čísly
    - proměnná, která je obsahuje, se nazývá ordinální (pořadová) - nemůžeme je sčítat, na rozdíl od kvantitativních (nelze počítat průměr ani jiné charakteristiky – také jako u kvalitativních proměnných)

Pro analýzu kvalitativních a ordinálních proměnných lze použít kvantitativní metody:

- lze zjišťovat **četnosti** výskytu jednotlivých kategorií (intervalů výdajů, typů výrobků)
- sledovat **závislost** dvou a více proměnných (zkoumat, zda interval ročních výdajů domácnosti na potraviny závisí na typu domácnosti)
- **aplikovat statistické metody** určené pro kombinace kvantitativních a kvalitativních proměnných (pro zkoumání, zda čas potřebný na vyhledání webových stránek závisí na typu internetového vyhledávače, nebo modelování možnosti domácnosti realizovat týdenní dovolenou mimo domov v závislosti na různých faktorech, např. na výši příjmů domácnosti a charakteristice osoby v čele)

Logika kvantitativního výzkumu je **deduktivní***, na začátku je problém existující buď v teorii nebo v realitě, tento problém je „přeložen“ do pracovních hypotéz – ty jsou pak základem pro výběr proměnných

**Deduktivní přístup vychází z positivismu, nejprve, na základě teorie, formuluje hypotézy a pomocí logického uvažování a již známých faktů testuje tuto hypotézu. K tomu využívá především kvantitativní data.*

Výstupem je potom soubor přijatých nebo zamítnutých hypotéz

Kvantitativní výzkum vyžaduje silnou **standardizaci**, která zajišťuje vysokou **reliabilitu**

Silná standardizace vede nutně k silné **redukci informace** – respondent místo toho, aby plně popsal svoje mínění (zkušenosti), je omezen na volbu jedné kategorie z nabídnutého, často velice malého, souboru kategorií -> vede k nízké validitě

Plusy:

- rychlý a přímočarý sběr dat a snadné zobecnění výsledků na celou populaci
- data jsou přesná, numerická a lehce ověřitelná
- výsledky jsou relativně nezávislé na výzkumníkovi

Mínusy:

- nebere v potaz lokální specifika, nepřichází s ničím novým, může pouze potvrdit či vyvrátit již zavedené teorie

Smíšený výzkum

=metodologická triangulace

- představuje kombinaci kvalitativních a kvantitativních metod při zkoumání
- v triangulaci jde o paralelní užívání různých druhů dat či různých metod při studiu jednoho a téhož problému

Tabulka 3.1: Rozdíly mezi kvalitativním a kvantitativním výzkumem

funkce/role	kvantitativní výzkum	kvalitativní výzkum
úloha výzkumu	přípravná	prostředek ke zkoumání reality
vztah výzkumníka k subjektu	neosobní (volný)	osobní (těsný)
postoj výzkumníka k jednání	vně situace	uvnitř situace
vztah teorie a výzkumu	potvrzení teorie	tvorba teorie
výzkumná strategie	silně strukturova-ná	slabě strukturovaná
šíře platnosti výsledků	monothetický	ideografický
data	tvrdá, široký záběr	měkká, jdou do hloubky
zaměření	makrosvět	mikrosvět

4. Základní protokoly internetu: IP, TCP, UDP, ICMP, DNS

Zkratka TCP/IP vznikla z názvů dvou protokolů: TCP (Transmission Control Protocol, tedy protokol řízení přenosů) a IP (Internet Protocol). **TCP/IP však představuje protokolovou sadu**, která obsahuje i další protokoly a standardy pro zasílání dat na sítích s přepínáním paketů. Technologie TCP zavádí virtuální spojení mezi dvojicemi koncových bodů, řídí přenos dat a zajišťuje jejich spolehlivé doručení. **Jednotlivé pakety se však zasílají prostřednictvím protokolu IP. Protokol IP se používá k zabalení dat zasílaných do sítě s přepínáním paketů. Kromě toho se zabývá také adresací systémů v síti.**

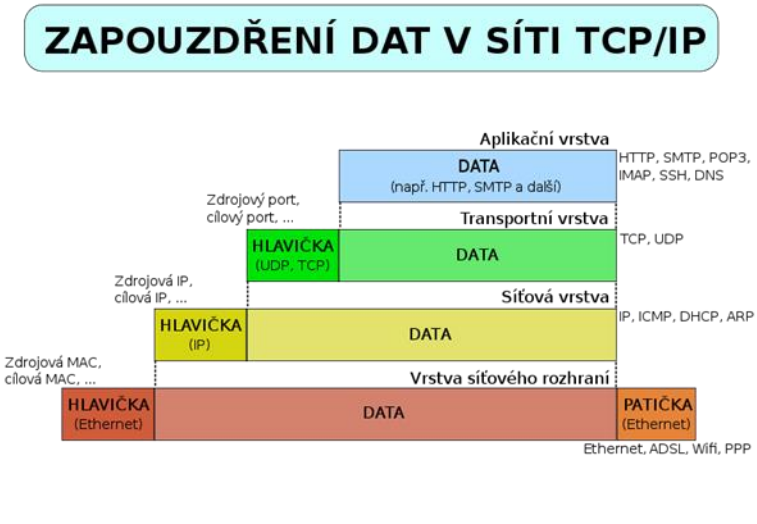
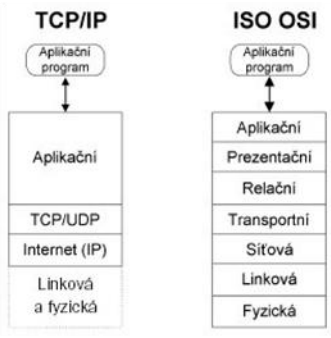


Schéma zapouzdření aplikačních dat na vrstvách ZCP/IP



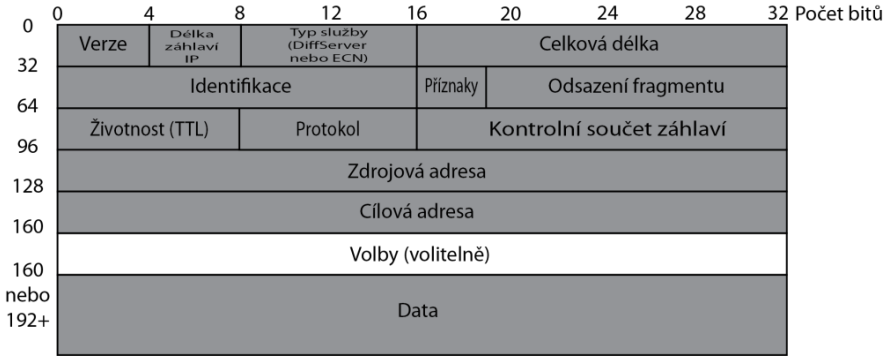
Porovnání modelů TCP/IP a ISO/OSI

Protokol IP

Internetový protokol IP je hlavní složkou sady protokolů, které **poskytují funkci doručování paketů** mezi koncovými body v **TCP/IP sítích**. Existují přitom dvě varianty IP: **IP verze 4** (IPv4), jež se dnes široce využívá, a dále protokol **IP verze 6** (IPv6), který se začíná zavádět do praxe. IP je protokol nezávislý na transportní vrstvě. Funguje v celé řadě různých sítí. **Byl navržen jako bezstavový, tolerantní k výpadkům a směrovatelný.**

Hlavička IP

Na začátek TCP nebo UDP dat je připojeno záhlaví IP, které se skládá z celé řady standardních polí. Obsahují identifikaci o zdroji a cíli paketu i o použitém formátu protokolu IP. Ze všech 13 polí je nutné všechna až na jediné (volitelná data) vyplnit.



Struktura záhlaví IP

- **Verze** – jedná se o 4bitovou hodnotu verze protokolu IP, buď 4 nebo 6.
- **Délka záhlaví IP** – toto pole odpovídá přítomnosti nebo naopak vynechání pole Volby. Uchovává celkovou délku záhlaví IP protokolu. V kombinaci s odsazením fragmentu se používá ke spolehlivé rekonstrukci datagramů.
- **Typ služby** – používá se ke specifikaci typu kvality služby (QoS). Aktuálně se v tomto poli mohou vyskytovat přiřazení rozlišených služeb (DiffServe) nebo explicitní notifikace o zahlcení (ECN). Uvádějí prioritu paketů v rámci IP a využívají se pro streaming multimediálních dat.
- **Celková délka** – definuje velikost datagramu včetně dat. Maximální možná hodnota je 216 (65 536) bajtů, minimální vyžadovaná délka je 576. Pokud je > 216 bajtů, datagramy se rozdělí na fragmenty.
- Identifikace – pole ID se používá ke specifikaci pořadí fragmentů v rozděleném IP datagramu.
- **Příznaky** – k dispozici jsou tři jednobitová pole pro nastavení příznaků. První má vždy hodnotu nula, další dvě souvisí s fragmentací paketů.
- **Odsazení fragmentu** – toto pole označuje místo v původní nefragmentované sekvenci dat, do něhož aktuální fragment má být zasazen. 1 -> nulové odsazení. Maximum 8192 fragmentů * 65 528 bajtů.
- **Životnost (TTL)** – 8bitové pole, které omezuje předávání paketu směrovačem nebo hostitelem. Maximální počet předání mezi směrovači (tzv. „hopy“). Hodnota TTL je dekrementována o 1 při každém předání. Když TTL = 0 -> směrovač paket se zahodí a pošle ICMP zprávu odesílateli.
- **Protokol** – uvnitř protokolu IP se může vyskytovat celá řada transportních protokolů. Toto 8bitové pole je od sebe odlišuje. Př.: 0 = IPv6 hop-by-hop, 1 = ICMP, 2 = IGMP (Internet Group Management Protocol), 6 = TCP, 17 = UDP, 27 = RDP(Reliable Datagram Protocol), 89 = OSPF, 129 = SMP(Simple Message Protocol) a 133 = FC(Fibre Channel)
- **Kontrolní součet záhlaví** – 16bitový kontrolní součet se ověřuje na každém směrovači po cestě. Pokud kontrolní součet nesouhlasí s vypočtenou hodnotou, paket je zahozen. Kontrolní součet se používá POUZE při předávání paketů. TCP, UDP a další transportní a aplikační protokoly mají vlastní kontrolní mechanismy, které zahrnují ověření platnosti také veškerých dat, jež dorazí do koncového bodu.
- **Zdrojová adresa** – zdrojová IPv4 adresa zapsaná v binární podobě.
- **Cílová adresa** – cílová IPv4 adresa koncového bodu, do něhož má paket dorazit.
- **Volby** – v tomto poli se mohou vyskytovat dodatečné informace, které mají nějaký vztah k paketu. Obsahuje změny ve zpracování paketu v rámci protokolu IP.
- **Data** – samotná datová část datagramu, která obsahuje „užitečný náklad“. Obsah dat se interpretuje na základě obsahu Protokol, nejčastěji se jedná o TCP nebo UDP data.

IPv4

První verzí internetového protokolu je IP verze 4. V současnosti se jedná o naprosto převažující standard. Protokol IPv4 je definován standardy IETF: RFC 791 a MIL-STD-1777.

Síťové adresy

Pro počítač se užívá homogenních 32 bitová čísel. Celý rozsah je teoreticky od 0.0.0.0 do 255.255.255.255, ale v tomto rozsahu je mnoho adres, které nemohou být adresy nějakého počítače. Tento zápis tak může nabývat celkem 2(4*8) hodnot. Celkově tedy dostáváme 4 294 967 296 možných adres. Problémem je vyčerpání prostoru IPv4 adres a řešením je přechod na protokol IPv6, jehož adresní prostor se pohybuje v rozsahu daném 128 bity. Celkově tedy existuje zhruba 3,4 x 1038 unikátních IPv6 adres.

Soukromé adresy

Ukázalo se, že mnohé organizace pro značnou část svých systémů nejenže nepotřebují, ale ani nechtějí globální IP adresy a globální konektivitu, přičemž chtějí a potřebují TCP/IP protokoly ve své organizaci. Pro tyto účely IANA rezervovala 3 bloky adres. Vytvoření těchto soukromých adres snížilo rychlost vyčerpávání globálních adres IP a vytvořilo v současnosti velmi silně používanou praxi překladů adres (NAT).

Rozsahy soukromých adres

Od	Do	Prefix
10.0.0.0	10.255.255.255	10/8
172.16.0.0	172.31.255.255	172.16/12
192.168.0.0	192.168.255.255	192.168/16

Třídy sítí

- A–8/24 (počet bitů sítě/počítače), 127 sítí, 16 milionů počítačů, počáteční bity: 0
- B–16/16 (počet bitů sítě/počítače), 16 tisíc sítí, 65 tisíc počítačů, počáteční bity: 10
- C–24/8 (počet bitů sítě/počítače), 2 miliony sítí, 254 počítačů, počáteční bity: 110

Pro organizace bylo málo sítí dle skupin, a tak se přešlo k beztřídnímu dělení IP adres (CIDR – classless internet domainrouting) pomocí masky sítě (stejné délky jako IP adresa – 4 bajty) počet jedniček určuje síť a nuly počítač, jedničky jsou vždy zleva.

Maska

Beztřídní dělení adres zobecňuje možnosti rozdělení adresy mezi část adresa sítě (network address) a část adresa systému (host address) ze tří možných hranic, které byly na hranicích jednotlivých bytů, na libovolnou hranici, která může být na libovolné bitové hranici. Určení, kde tato hranice bude, se děje objektem nazývaným síťová maska (network mask, subnet mask).

Dělení sítě s prefixem /24 na podsítě

Poslední bajt v desítkovém zápisu s tečkami	Poslední bajt binárně	Počet unikátních adres pro stanice[i]	Počet možných podsítí	Efektivní prefix CIDR
255	11111111	nelze	nelze	/32
254	11111110	2 (point-to-point)	128	/31
252	11111100	2	64	/30
248	11111000	6	32	/29
240	11110000	14	16	/28
224	11100000	30	8	/27
192	11000000	62	4	/26
128	10000000	126	2	/25
0	00000000 (žádná maska)	254	1 (žádná podsíť)	/24

IPv6

IPv6 je druhou verzí internetového protokolu IP. Je nástupcem IPv4 a jedním z hlavních problémů, které má IPv6 vyřešit, je **poskytnutí výrazně většího adresního prostoru**, dále **lepší granularita** (zcela automatická konfigurace a zdokonalení směrování) a **zlepšení zabezpečení**.

IPv6 řeší spoustu problémů, které vznikly problematickým návrhem IPv4. Záhlaví IPv6 je jednodušší a má zabudovaný přirozený mechanismus pro vyřešení kvality služby (QoS) známý jako **štítkování toků**. Neuvěřitelně obrovský adresní prostor dovoluje zcela **zapomenout na podsítě**, které již nejsou vůbec nutné, stejně tak i **NAT již není potřeba**. Problémy s překladem adres mají hlasové protokoly VoIP, BitTorrent, SIP (Session Initiation Protocol) a další protokoly pro streaming nebo P2P.

Protokol IPv6 NDP (Neighbor Discovery Protocol) nahrazuje IPv4 protokol ARP. NDP dokáže nejen pomoci s rozpoznáváním okolních stanic a zařízení, ale pomáhá také s identifikací síťových prefixů a správné metody **automatické konfigurace sítě**.

Transportní vrstva je 4. vrstva modelu vrstvové síťové architektury (OSI). V originále se nazývá transport layer. Poskytuje transparentní, spolehlivý přenos dat s požadovanou kvalitou. Vyrovnává různé vlastnosti a kvalitu přenosových sítí. Provádí převod transportních adres na síťové, ale nestará se o směrování. Tato vrstva zajišťuje přenos dat mezi koncovými uzly. Jejím účelem je poskytnout takovou kvalitu přenosu, jakou požadují vyšší vrstvy. Principiálně nabízí tato vrstva dva typy služeb – spojově (TCP) nebo nespojově orientované (UDP).

Základem komunikace mezi vzdálenými počítači, tedy síťové komunikace, je model TCP/IP. Síťová komunikace je rozdělena do tzv. vrstev, které znázorňují hierarchii činností. Každá vrstva využívá služeb vrstvy nižší a poskytuje své služby vrstvě vyšší. 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. Příkladem může být možnost komunikace po různých fyzických médiích – Ethernet, token ring, sériová linka.

TCP (Transmission Control Protocol)

TCP je spojově orientovaný protokol. Spojení může otevřít klient nebo server a pak mohou už být posílána jakákoliv data oběma směry. Charakteristické vlastnosti TCP protokolu jsou:

- spolehlivost – TCP používá potvrzování o přijetí, opětovné posílání a překročení časového limitu. Pokud se jakákoliv data ztratí po cestě, server si je opětovně vyžádá. U TCP nejsou žádná ztracená data, jen pokud několikrát po sobě vyprší časový limit, tak je celé spojení ukončeno.
- zachování pořadí – jestliže se odešlou 2 zprávy, jedna po druhé, první dorazí nejdřív k serveru. Pokud data dorazí ve špatném pořadí, TCP vrstva se postará o to, aby některá data pozdržela a finálně je předala správně seřazená.
- vyšší režie – TCP protokol potřebuje tři pakety jen pro otevření spojení, což však umožňuje zaručit spolehlivost celého spojení.
- Protokol TCP dopravuje data mezi dvěma konkrétními aplikacemi, na rozdíl od IP protokolu, který dopravuje data jen mezi dvěma počítači, proto se v případě TCP protokolu přidává port (0-65535), aby se dalo odlišit, pro jakou aplikaci data jsou. Několik příkladů: FTP (port 21 a 20), SMTP (port 25), DNS (port 53) a HTTP (port 80).
- Vytváří spojení mezi aplikacemi a na dobu propojení vytváří virtuální okruh (plně duplexní → obousměrná komunikace)
- Integrita dat je zajištěna kontrolním součtem → ochrana pouze proti modifikaci dat při přenosu (ne proti inteligentním útočníkům, na to jsou protokoly např. SSL, S/MIME)

Aplikace posílá proud (stream) bajtů TCP protokolu k doručení sítí, TCP rozděluje proud bajtů do přiměřeně velkých segmentů. TCP pak předá takto vzniklé pakety IP protokolu k přepravě internetem do TCP modulu na druhé straně spojení. TCP ověří, že se pakety neztratily tím, že každému paketu přidělil číslo sekvence, které se také použije k

ověření, že data byla přijata ve správném pořadí. TCP modul na straně příjemce posílá zpět potvrzení pro pakety, které byly úspěšně přijaty. Pokud by se odesilateli potvrzení nevrátilo do určité doby, vypršel by odesilatelův časovač a (pravděpodobně ztracená) data by byla vyslána znovu.

Navázání spojení

Otevření je třícestný handshaking. První strana otevře tzv. socket, vytvoří možnost komunikace (pasivní open).

1. Druhá strana požádá o otevření spojení zasláním SYN paketu
2. První strana odpoví paketem SYN/ACK
3. Druhá uzavře dohodu paketem ACK.

TCP přenos dat

- Během navázání spojení je vyměněno ISN mezi účastníky spojení. A dohodnuto o velikosti okna.
- Při přenosu dat je zasíláno SN a v poli ACK je zasíláno potvrzující SN které bylo přijato.
- Je počítána 16bitová checksum z dat. → Chybné pakety zahozeny.

Ukončení spojení

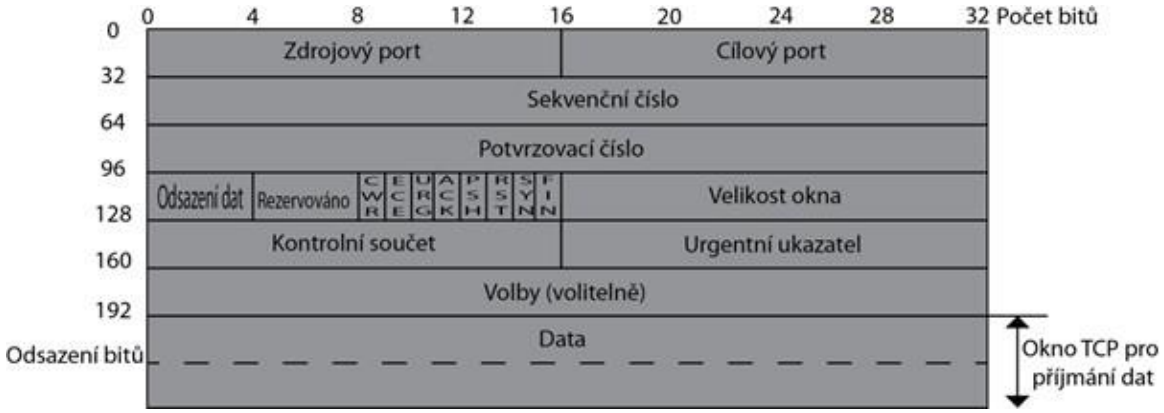
Ukončení je opět třícestný handshaking

1. Jedna strana požádá o uzavření spojení zasláním FIN paketu
2. Druhá strana odpoví paketem FIN/ACK
3. První uzavře spojení paketem ACK.

Hlavička TCP

- Zdrojový port – 16bitů – port odesílatel segmentu
- Cílový port – 16bitů – port příjemce segmentu
- Sekvenční číslo (SN) – 32bitů – pořadové číslo prvního bajtu v segmentu (pokud není nastaven příznak SYN). Pokud je nastaven příznak SYN, jedná se o tzv. initialsequencenumber – ISN a první datový oktet má číslo ISN + 1
- Potvrzovací číslo (AN) – 32bitů – číslo následujícího bajtu, který je příjemce připraven přijmout, tzn. do toho čísla všechny bajty v pořádku přijal
- Odsazení dat – 4bity – délka záhlaví TCP segmentu ve 4bajtech (jako u IP datagramu)
- Rezerva – 4bity– mělo by být nulové
- Příznak – kontrolní bity (příznaky) zajišťující „handshaking“ a ostatní specifické procesy
    - URG – Urgent Pointer
    - ACK – Acknowledgement
    - PSH – Push funkce
    - RST – Reset spojení
    - SYN – synchronizace sekvenčních čísel
    - FIN – oznámení, že odesílající nemá žádná další data
- Velikost okna – vyjadřuje přírůstek pořadového čísla bajtu, který bude ještě příjemce akceptovat
- Kontrolní součet – TCP kontrolní součet se počítá z přenášených dat+TCP hlavičky+ některé položky IP hlavičky. Vždy sudý počet bajtů, pokud ne, data se fiktivně doplní jedním bajtem na konci
- Urgentní ukazatel – lze jen v případě příznaku URG, přičte-li se ukazatel k pořadovému číslu odesílaného bajtu, pak ukazuje toto číslo na konec naléhavých dat. Využívá např. telnet

- Odsazení bitů – jako u IP datagramu – výplň do násobku 4bajtů



Struktura TCP paketu spolu se všemi sekcemi záhlaví

Užití

Mail, web a obecně služby, kde je třeba bezdrátový přenos dat

UDP (User Datagram Protocol)

UDP je jednodušší protokol založený na odesílání nezávislých zpráv. Jedná se o nespojovanou službu, takže datagram je odeslán a odesilatel se už nestará, zda dorazil k příjemci (o to se stará aplikační protokol).

Charakteristika protokolu:

- bez záruky – protokol už neumožňuje ověřit, jestli data došla zamýšlenému příjemci. Datagram se může po cestě ztratit. UDP nemá žádné potvrzování, přeposílání ani časové limity.
- nezachovává pořadí – jestliže odešleme dvě zprávy jednomu příjemci, nemůžeme předvídat, v jakém pořadí budou doručeny.
- jednoduchost – nižší režie, než u TCP (není zde řazení, žádné sledování spojení atd.).

Hlavička

- Zdrojový port – 16bitů – port odesílatel segmentu
- Cílový port – 16bitů – port příjemce segmentu
- Délka dat – 16 bitů – (délka hlavičky + dat), min je 8bajtů (délka hlavičky)
- Kontrolní součet – nepovinný, jinak jako u TCP segmentu (hlavička + data + něco z IP hlavičky), vždy sudý počet bajtů, pokud ne, data se fiktivně doplní jedním bajtem na konci

Struktura UDP datagramu

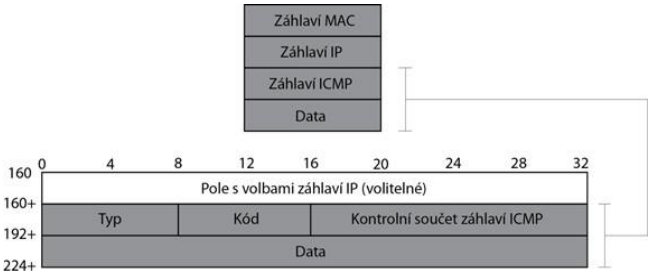
Užití

Pro aplikace pracující systémem otázka-odpověď (např. DNS, sdílení souborů v LAN). Jeho bezstavovost je užitečná pro servery, které obsluhují mnoho klientů, nebo pro nasazení, kde se počítá se ztrátami datagramů a není vhodné, aby se ztrácel čas novým odesíláním (starých) nedoručených zpráv (např. VoIP, video stream, apod.)

ICMP

- Služební protokol, podmnožinou IP protokolu, ale chová se stejně jako TCP/UDP protokol, vkládá se do IP datagramu (IP hlavička, ICMP hlavička, data)
- Slouží k signalizaci mimořádných událostí v sítích založených na IP protokolu
- Nepoužívá se většinou přímo aplikacemi, ale jsou výjimky jako např. ping (echo request -> echo reply)
- Například každý směrovač, který přeposílá IP datagram, musí v IP hlavičce dekrementovat pole TTL („time to live“) o jedničku. Jestliže TTL klesne na 0 (a datagram není určen stroji provádějícímu dekrementaci), router přijatý paket zahodí a původnímu odesilateli datagramu pošle ICMP zprávu „Time to live exceeded in transit“.
- Některé zprávy jsou např.: Nedoručitelný IP datagram, Snížit rychlost odesílání, Změň směrování, Žádost o echo, Echo, Odpověď na žádost o směrování, čas vypršel, chybný parametr atd.
- Pro správné fungování protokolu IP musí systémy podporovat zprávy ICMP. Korektní chování ICMP je nutnou podmínkou pro bezchybnou komunikaci v sítích IP. Existují přitom dvě verze tohoto protokolu, jedna určená pro IPv4 a druhá pro IPv6.
- ICMP se svým účelem liší od TCP a UDP protokolů tím, že se obvykle nepoužívá síťovými aplikacemi přímo. Výjimkou je např. nástroj ping, který posílá ICMP zprávy „Echo Request“ (a očekává příjem zprávy „Echo Reply“), aby určil, zda je cílový počítač dosažitelný a jak dlouho paketům trvá, než se dostanou k cíli a zpět.
- Zprávy ICMP se generují na základě datagramů IP, jejichž vlastností nebo chování si akci na úrovni protokolu ICMP vyžaduje. ICMP zprávy se nikdy nedělí a jsou vždy v jediném datagramu, není proto nutné provádět složité ověřování jako v TCP. Znamená to, že ICMP je nespolehlivý formát přenosu. Typ ICMP zprávy se rozpoznává podle prvního pole vloženého na pozici 160 v paketu (ihned za záhlaví IP), případně dále, pokud by bylo v záhlaví IP zaplněno pole s volbami. Strukturu záhlaví ICMP zprávy znázorňuje obrázek níže.

Struktura záhlaví ICMP



DNS – Domain Name Server

DNS je protokol, jehož hlavní funkcionalita spočívá v **překládání názvů domén na IP adresy**. Systém DNS je hierarchický, každý **DNS server se stará o svou doménu** (nebo několik domén) a může delegovat správu DNS svých subdomén na podřízené DNS servery. Překlad IP x.x.x.x na www.xxx.cz

Původně DNS řídil jen překlad názvů domén na IP adresy, dnes se ale **používá i pro další služby**, např. udržuje **informace o poštovních serverech domény** (tzv. **MX záznamy**).

Adresu DNS serverů musí znát každý klient v síti, jinak nemůže používat URL adresy. V praxi to znamená, že URL „seznam.cz“ je pro cestu v síti třeba přeložit na IP adresu. Adresu DNS serverů má klient v síti nastavenou buď **staticky**, nebo jí **získá spolu s IP adresou od DHCP serveru**.

DNS je protokol aplikační vrstvy, stejně jako DHCP.

Hlavní elementy DNS

- DNS resolver – klient k DNS systému, běží na uživatelském počítači a generuje dotazy na požadavky programů.
- Rekurzivní DNS server – přijme dotaz a hledá v DNS odpověď
- Autoritativní DNS server – odpovídá rekurzívním dotazům buď ze svých informací nebo odkazem

DNS rekurze

- Recursor musí znát Root servery.
- Zeptá se root serveru na odpověď
- Dostane odpověď ne přímo, ale ať se recursor zeptá na určité adrese, že tam je odpověď domény první úrovně.

DNS cache poisoning

- jde o zranitelnost DNS protokolu, kdy lze DNS serveru podvrhnout odpověď s falešnými IP adresami
- DNS server si odpověď nacachuje a pak ji distribuuje dalším klientům
- Řešením je nasazení technologie DNSSEC, kdy autentičnost odpovědi bude zajištěna elektronickými podpisy a ověřováním jejich platnosti (viz http://www.dnssec.cz/)

5. Aplikační nástroje internetu: email, ssh, WWW

Elektronická pošta je způsob odesílání, doručování a přijímání zpráv přes elektronické komunikační systémy. Termín **e-mail** se používá jak pro internetový systém elektronické pošty založený na protokolu **SMTP** (Simple Mail Transfer Protocol), tak i pro intranetové systémy, které dovolují posílat si vzájemně zprávy uživatelům uvnitř jedné společnosti nebo organizace (tyto systémy často používají nestandardní protokoly, mívají ovšem bránu, která jim dovoluje posílat a přijímat e-maily z internetu). K širokému rozšíření e-mailu přispěl zejména Internet.

Mezi počítači na Internetu se vyměňují zprávy pomocí SMTP a softwaru typu MTA (Mail Transfer Agent) jako např. Sendmail.

Uživatelé mívají na svém počítači nainstalován program, který se nazývá **e-mailový klient**. Ten stahuje zprávy z poštovního serveru použitím protokolů **POP** nebo **IMAP**, avšak v prostředí velkých společností se stále vyskytuje použití některého komerčního protokolu jako např. **Lotus Notes** nebo **Microsoft Exchange Server**. Někteří uživatelé nepoužívají e-mailového klienta, ale přistupují ke zprávám umístěným na poštovním serveru přes **webové rozhraní**. Tento postup se často používá zejména u **freemailových** (bezplatných) služeb.

Je možné ukládat e-maily buď na straně serveru, nebo na straně klienta. Standardní formáty pro mailové schránky jsou např. Maildir a mbox. Několik e-mailových klientů používá vlastní formát a na konverzaci mezi těmito formáty je potřebný speciální program.

Ze základních funkcí Internetu je e-mail jedinou funkcí, kde (hlavní) přenos probíhá v pozadí. Uživatel obvykle jen přímo naedituje (napíše) text dopisu a ten je pak v pozadí nějak v Internetu dopraven až k adresátovi. Ovšem interaktivní spolupráci bezprostředně dvou spolu komunikujících programů se ani v tomto případě nevyhneme. Takto spolu totiž (bez naší účasti) komunikují dva automaticky pracující programy. Způsob jejich spolupráce popisuje protokol SMTP. Programy se přitom musí umět nějak vyrovnat i s problémy, které souvisejí buď s dočasným výpadkem vzdáleného počítače, anebo s přetížením přenosové cesty.

Doručování

Při posílání pošty přes internet má být zaručen spolehlivý přenos zprávy i v případě dočasného výpadku cílového serveru.

1. Zpráva se obvykle píše v prostředí programu typu **e-mailového klienta** nebo v obdobném formuláři webového rozhraní.
2. **Klient** pomocí **Simple Mail Transfer Protocol (SMTP)** pošle zprávu **programu Mail Transfer Agent (MTA)**, například smtp.a.org, který může běžet buď na samostatném smtp poštovním serveru, nebo i přímo na počítači odesílatele.
3. **Program MTA** zjistí z uvedených cílových adres název domény (část adresy za zavináčem) a tyto domény vyhledá v **Domain Name System (DNS**), aby zjistil **mail exchange servery** přijímající poštu pro danou doménu.
4. **DNS server** domény b.org, tedy ns.b.org, **odpoví MX záznamem**, kde uvede mail exchange server pro danou doménu.
5. **MTA server** (např. smtp.a.org) odešle zprávu na **mail exchange server** (např. mx.b.org) pomocí protokolu SMTP. Domény obvykle mají záložní (backup) mail exchange server, takže můžou pokračovat v přijímaní pošty, i když je právě nedostupný hlavní mail exchange server.

   Když není možné zprávu doručit, MTA příjemce o tom musí odeslat zpět odesílateli zprávu (en:bounce message), ve které ukazuje na problém.

6. Mail exchange server zprávu doručí do schránky adresáta.
7. Ze schránky adresáta si zprávu stáhne pomocí protokolu **POP (POP3)** nebo **IMAP** nebo ji adresátovi umožní prohlédnout poštovní klient příjemce nebo webová služba.

Bývalo zvykem, že kterýkoliv MTA přijímal zprávy pro kteréhokoli uživatele na Internetu a udělal, co se dalo, aby zprávu doručil. Takové MTA se nazývají **en:open mail relays**. To bylo důležité v začátcích Internetu, když byla síťová spojení nespolehlivá a nepermanentní. Když MTA nemohl doručit zprávu do cíle, mohl ji alespoň poslat agentovi bližšímu k cíli. Ten by měl větší šanci ji doručit. Ukázalo se však, že tento mechanizmus byl zneužitelný lidmi posílající nevyžádanou hromadnou poštu (SPAM), a proto velmi málo ze současných MTA jsou open mail relays (tj. přijímají poštu pro známé uživatele, resp. domény). Prakticky všechny open relays jsou rychle odhalené a zneužité spamery, kteří neustále prohledávají (skenují) IP rozsahy celého internetu.

Kódování obsahu e-mailu

Pro e-mail je definován přenos 7bitové ASCII informace. Přesto je většina e-mailových přenosů 8bitových, kde ale nelze zaručit bezproblémovost. Z toho důvodu byla elektronická pošta rozšířena o standard MIME, aby bylo umožněno kódování vkládaných HTML a binárních příloh, obrázků, zvuků a videí.

SMTP

Celé pojetí SMTP vychází z představy Internetu jako multiuživatelských počítačů trvale spojených komunikačními linkami. Na počítači, kde uživatel pracuje, je stále v provozu jistý program označený třeba SMTP, který se stará o odesílání a přijímání dopisů. Hlavní zásada protokolu SMTP spočívá v tom, že spolu bezprostředně komunikují programy na počítači odesílatele a počítači adresáta.

1. Spolupráce je spojovaná, přenosový kanál na portu 25 otevírá program, který má dopis odeslat.
2. Po otevření kanálu se program, který na cílovém počítači dopis přijímá, hned představí.
3. Odesílající program pak začíná seanci typickým hello a uvedením jména svého počítače.
4. Po úspěšném potvrzení pak uvede E-mailovou adresu odesílatele, po jejím potvrzení adresu příjemce, tu především mu musí cílový program potvrdit, zda vůbec takového příjemce (uživatele) zná.
5. A potom po akceptování příkazu data předává řádek po řádku text celého dopisu.
6. Text dopisu končí (obdobně jako tomu je u Gopheru) samotnou tečkou na celém řádku.
7. Přijímající program se tak navrátí do příkazového režimu a další zasílané řádky opět chápe jako příkazy SMTP-konverzace.
8. Příkazem quit pak může být z odesílajícího programu celá seance ukončena. Anebo lze hned posílat jiný dopis (jiného uživatele) určený příjemci na stejném počítači. Dokonce bývalo i možné, aby nyní program začal naopak přijímat z cílového počítače poštu pro své uživatele.

Příklad

Connected to adis.cesnet.cz.

220 adis.cesnet.cz ESMTP Sendmail 8.8.8/CESNET; Tue, 29 Jun 1999 12:44:08 +0200 (MET DST)

hello beba.cesnet.cz

250 adis.cesnet.cz Hello beba [194.50.6.2], pleased to meet you

mail from:erijk@beba.cesnet.cz

250 <erijk@beba.cesnet.cz>... Sender ok

rcpt to:<listserv@adis.cesnet.cz>

250 <listserv@adis.cesnet.cz>... Recipient ok

data

354 Enter mail, end with "." on a line by itself

help

list

.

250 MAA18057 Message accepted for delivery

quit

221 adis.cesnet.cz closing connection

Connection closed by foreign host.

Dopis byl přenášen mezi počítači beba.cesnet.cz a adis.cesnet.cz. Odesílatelem byl uživatel erijk@beba.cesnet.cz a určen byl příjemci listserv@adis.cesnet.cz. Odpovědi přijímajícího programu na adis.cesnet.cz vždy začínají trojciferným číslem, které je pro další činnost podstatné. Slovní text je jen vysvětlující, především pro objasnění případných problémů s přenosem uživatelům E-mailu. Trojciferné číslování odpovědí je konstruováno tak, že cifrou **2**

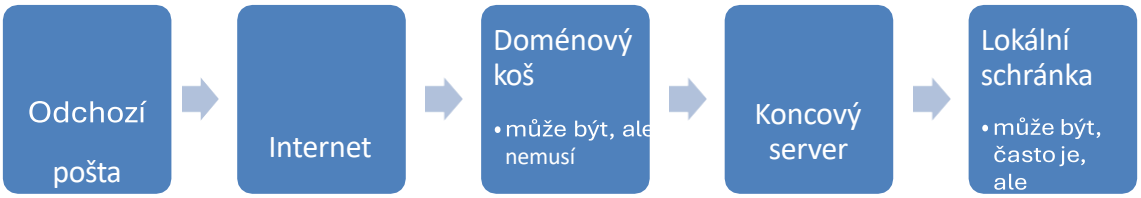
začínají odpovědi související s činností úspěšně provedenou, **3** začíná jediná odpověď **354** oznamující přechod do datového režimu, **4** je určena pro chyby, které mohou být po čase (nějak) odstraněny (např. **452** – nedostatek místa na disku), **5** zahrnuje podstatné, neodstranitelné chyby při doručování dopisu (např**. 550** – neznámý uživatel).

Příchod osobních počítačů, které se vypínají, a neslouží jen jako dokonalejší vzdálený terminál takovýchto počítačů, a v neposlední řadě také mnohá současná omezení pro vytváření přenosových kanálů jen na některých povolených portech pak znamenaly nutný posun v této koncepci. SMTPprogram odesílající dopis nemusí nutně kontaktovat program na počítači, jehož jméno je napsáno v Emailové adrese konečného příjemce. Na domain-name serveru, kam se většinou stejně musí obrátit pro zjištění IP-adresy (kvůli vytvoření přenosového kanálu), může být pro počítačové jméno adresy příjemce uveden tak zvaný **MX-record**, neboli informace o Mail Exchangeru. Ta pak může říkat, že pro přijímání pošty je určen počítač jiný, na který se má dopis předávat (pomocí SMTP-protokolu). Kromě toho zde mohou být uvedeny ještě další informace, například též náhradní počítač, kam je možno dopis předat**. V praxi vypadá DNS záznam například: „jmeno_firmy.cz, TTL 1800, TYP MX, Hodnota 10 mxavas.isp.cz“** Je pak věcí koncového adresáta, zda ví, že svou poštu musí vždy číst na některém jiném počítači, anebo (jak zpravidla bývá uděláno) mu pošta přijde už třeba pomocí jiného protokolu až na počítač, který běžně používá.

Doménový koš

Primární server domény nemusí být vždy server, na němž jsou umístěny schránky lokálních uživatelů. Je-li např. firma, které daná doména patří, připojena do Internetu vytáčenou linkou, může mít u poskytovatele Internetu zřízen tzv. doménový koš, což je jedna schránka, do níž se ukládá pošta pro celou doménu. Koncový poštovní server pak může tuto schránku vybírat (v časech či intervalech dle potřeby) a poštu třídit do jednotlivých lokálních schránek. V MX záznamech pro tuto doménu je pak jako primární server uveden SMTP server poskytovatele, u něhož je doménový koš umístěn.

Doménový koš přijímá zprávy z Internetu přes SMTP protokol. Každá zpráva tedy obsahuje jak tělo, tak i SMTP obálku. Doménový koš přijímá pouze tělo zprávy. Informace z obálky se zkopírují do některé z hlaviček zprávy (záleží na nastavení doménového koše).



Protokoly pro stahování pošty

POP3

POP3 (Post Office Protocol version 3) je internetový protokol navržený pro stahování pošty ze serveru na jiný server (viz sekci Doménový Koš) nebo na poštovního klienta. POP3 protokol je definován v RFC 1939.

POP3 protokol pracuje na **bázi klient-server**. Komunikaci vždy navazuje klient, dále se dotazy klienta a odpovědi serveru pravidelně střídají, dokud nedojde k ukončení komunikace. Jakmile klient inicializuje komunikaci a provede autentizaci (ověření jménem a heslem), je možné s poštou začít pracovat (stahovat ji na klienta, mazat ji, a podobně).

POP3 protokol je dnes již poněkud jednoduchý a zastaralý. **Umí v podstatě pouze stáhnout poštu do klientské aplikace, umí pracovat pouze s jednou složkou (INBOX).** To znamená, že pokud uživatel přesune v klientovi zprávu do jiné složky, bude tato zpráva ze serveru přesunuta (de facto smazána). Obdobně se mailbox chová i v opačném případě. Pokud uživatel má na serveru přístup k více složkám a zprávu z Inboxu přesune do jiné složky, tak se zpráva do klientské aplikace nemůže stáhnout. Obecně lze říci, že je doporučováno využívat pro práci s poštou **modernější protokol IMAP**.

Jedinou výhodou použití tohoto protokolu může být **ušetření místa na disku serveru**. Uživatelé si svou poštu stáhnou lokálně na disk a zde si ji mohou roztřídit do složek, mazat ji atd. Z toho důvodu jsou **POP3 účty využívány zejména v případě freemailových služeb**, kde mohou uživatelé využívat schránku o několika MB a poštu si více méně pravidelně stahují lokálně na svůj disk. Další výhodou může být snadná možnost **práce offline**, kterou lze využít v případě časově omezeného připojení do Internetu. Dnes však již téměř všichni poštovní klienti umí pracovat v režimu offline jak s účty typu POP3, tak s IMAP účty.

IMAP

IMAP (Internet Mail Access Protocol) je internetový protokol používaný, stejně jako protokol POP3, pro připojení k poštovnímu serveru a čtení nebo jiné manipulaci s poštou. IMAP protokol je definován v RFC 3501.

Protokol IMAP umožňuje uživatelům nejen **stahování pošty** na svůj počítač (do svého e-mailového klienta), ale také **správu účtu přímo na serveru**. Díky tomu je možné k účtu přistupovat **z různých klientských stanic**. Na rozdíl od protokolu POP3 **nechává protokol IMAP poštu na serveru** a přímo tam lze zprávy s pomocí poštovního klienta číst, rušit a ukládat do nově vytvořených složek, tak jako by byla schránka uložena přímo na disku klienta. Zároveň je možné mít poštu uloženou v poštovním klientovi. Toto řešení je žádoucí především tehdy, pokud uživatel používá časově omezené připojení do Internetu, nebo je z jiných důvodů připojen k serveru jen někdy a je nutné mít k dispozici svou poštu offline. Po opětovném připojení k síti se složky na serveru a klientovi synchronizují.

Dalším podstatným rozdílem mezi protokoly POP3 a IMAP je **možnost manipulace se zprávami již během stahování pošty** do lokálního úložiště. V případě protokolu IMAP se totiž nejprve stáhnou hlavičky e-mailů a uživatel si tak může vybrat, který si chce přečíst jako první. Po označení vybrané zprávy dostane tato zpráva prioritu ve stahování na klienta a je možné ji přečíst nebo přesunout do jiné složky nebo cokoliv jiného, zatímco jsou stahovány ostatní zprávy.

Architektura pošty

Doručování elektronické pošty po Internetu se účastní tři druhy programů:

- **MUA - Mail User Agent**, poštovní klient, který zpracovává zprávy u uživatele
- **MTA - Mail Transport Agent**, server, který se stará o doručování zprávy na cílový systém adresáta
- **MDA - Mail Delivery Agent**, program pro lokální doručování, který umísťuje zprávy do uživatelských schránek, případně je může přímo automaticky zpracovávat (ukládat přílohy, odpovídat, spouštět různé aplikace pro zpracování apod.)

Poštovní klient (MUA) je program, který **zajišťuje odesílání zpráv a vybírání schránek**. Příkladem je např. Microsoft Outlook, Mozilla Thunderbird, Opera, Mutt, Pine a další. Je to v podstatě specializovaný editor, který umí kromě vytvoření zprávy také manipulovat se schránkami, odeslat zprávu nejbližšímu MTA a převzít zprávu ze serveru prostřednictvím POP3 nebo IMAP. Vlastním doručováním zprávy po síti až k adresátovi se klient nezabývá. Součástí klienta bývá také více či méně složitý adresář, který pomáhá uživateli udržet přehled o adresách.

Poštovní server (MTA) běží obvykle jako démon a naslouchá na portu **TCP/25**. K tomuto portu se může připojit (navázat TCP spojení) buď poštovní klient, nebo jiný server, který předá zprávu k doručení. MTA zkontroluje, zda je zpráva určena pro systém, na kterém běží. Pokud ano, předá ji programu MDA (lokální doručení). Pokud je zpráva určena jinému počítači, naváže spojení s příslušným serverem a zprávu mu předá.

Při vyhledávání vzdáleného serveru, kterému má předat zprávu, musí MTA spolupracovat se systémem DNS. Od serveru DNS si vyžádá tzv. MX záznam pro cílovou doménu, který obsahuje IP adresu počítače, který se stará o doručení pošty v této doméně.

Poštovní server obsahuje v konfiguraci řadu parametrů, pomocí kterých můžeme mimo jiné nastavit, pro které domény MTA přijímá zprávy. Stejně tak je možné určit, od koho bude nebo nebude zprávy přijímat, což je velmi důležité z hlediska bezpečnosti a ochrany proti spamu.

Nejčastějšími programy v roli MTA jsou sendmail, postfix, qmail, Microsoft Exchange, Mercury aj.

SSH

SSH (Secure Shell) je program a síťový protokol v sítích TCP/IP umožňující bezpečnou komunikaci mezi dvěma počítači pomocí **transparentního šifrování a volitelné komprimaci přenášených dat.** Pracuje na portu 22. Pokrývá tři základní oblasti bezpečné komunikace: autentizace obou účastníků komunikace, šifrování a integritu přenášených dat, které jsou přenášeny mezi dvěma počítači přes nebezpečnou vnější síť. Umožňuje volitelnou komprimaci přenášených dat

SSH je ve počítačové terminologii používán jako **název přenosového (síťového) protokolu** i jako název programu zprostředkovávající spojení. Označení „Secure Shell" je mírně zavádějící, protože **nejde ve skutečnosti o shell ve smyslu interpret příkazů**. Název byl odvozen z existujícího programu rsh, který má podobné funkce, ale není zabezpečený.

SSH program je dnes běžně používán při vzdálené práci a pro vzdálenou správu. Většinou se **spojuje s SSH démonem** (SSH daemon, sshd) pro navázání spojení. SSH démon rozhoduje podle svého nastavení**, zda spojení přijme**, jakou **formu autentizace** bude požadovat, případně na **kterém portu naslouchá**. Implementace SSH klientů i serverů (SSH démon) je dostupná na téměř jakékoli platformě. Většinou jsou dostupné jak komerční, tak i Open Source varianty. Jedná se také o nejčastější způsob vzdáleného připojení na UNIX/Linux systémy, neboť právě ty často fungují pouze s příkazovým řádkem.

- Poskytuje komunikaci šifrovaným kanálem po TCP/IP síti
- Navržen jako náhrada telnetu a dalších, které posílají heslo v nezabezpečené formě (umožnění odposlechu hesla)
- Přístup k příkazovému řádku, ale i přenos souborů nebo jakýkoliv datový přenos s využitím síťového tunelování
- Zabezpečuje autentizaci obou účastníků komunikace, šifrování přenášených dat a jejich integritu, a i volitelnou bezztrátovou kompresi
- Server naslouchá na portu 22

Vrstvy:

Transportní

- Vytvoření spojení, integrita
- Zajištění, že se strany komunikace v průběhu nezměnily (opakovaná výměna klíčů po přenesení určitého množství dat nebo po uplynutí určité doby)

Autentizační

Ověření identity uživatele různými způsoby:

- Heslem
- Veřejným klíčem
    - např. DSA, RSA o privátní klíč má klient, veřejný klíč je na serveru (např. v ~/.ssh/authorized_keys)
    - server veřejným klíčem zašifruje blok náhodných dat a pošle ho klientovi
    - klient privátním klíčem blok rozšifruje a pošle data zpět serveru
    - pokud zpět zaslaná data odpovídají původním, je autentizace úspěšná
- Keyboard-interactive
    - uživatel zadá heslo nebo jiné požadované informace pro autentizaci na klávesnici a tento vstup je odesílán serveru
- GASSAPI
    - Umožňuje přihlášení pomocí externích mechanismů, jako jsou systémy pro centrální autentizaci (např. Kerberos)

Spojovací

- Definuje kanály spojení, přes které jsou poskytovány SSH služby
- Kanálů může být při jednom spojení aktivních několik a kanály mohou přenášet data v obou směrech
- Typy kanálů:
    - Shell – Terminál (vzdálené spuštění programu), přenos souborů (SFTP nebo SCP)
    - X11 – X Window System – přenos grafického uživatelského prostředí
    - Forwarded-tcpip – Přesměrovávání (tunelování) ve směru klient-server
    - Direct-tcpip – Přesměrovávání (tunelování) ve směru server-klient

WWW (world wide web)

- Služba WWW vznikla na půdě CERNu v letech 1989-90 pod vedením TimaBerners-Leeho. Měla sloužit jako infrastruktura pro sdílení výsledků vědeckých výzkumů.
- Je to informační prostor, v němž jsou jeho položky interpretované jako zdroje, jsou identifikovány globálními identifikátory URI
- Základy a principy WWW
    - Lokace zdrojů (URL a URI)
    - Přenos informací (http protokol)
    - Reprezentace informací (HTML jazyky)

Tři Pilíře WWW

- **Identifikace**: Každý zdroj je identifikován jedinečným URI (URL)
- **Interakce:** web agent komunikuje standardním protokolem, který provádí interakci výměnou zpráv dané syntaxe a sémantiky (HTTP)
- **Interpretace (reprezentace):** V protokolu přenášíme data a popřípadě metadata z určeného zdroje (HTML)

URI (Identifikace – 1. pilíř)

- **Identifikace:**
    - URI vytváří globální pojmový prostor a tím globální síťový efekt.
    - Idea pochází z roku 1990 od DouglaseEngelbarta.
- řetězec znaků s definovanou strukturou, který slouží ke specifikaci zdroje informací (dokument/služba) zejména pro použití v rámci internetu
- URL – UniformResourceLocator
    - definuje doménu, umístění na serveru a protokol, port, parametry
- URN – UniformresourceName
    - definuje zdroj, ale nesnaží se popsat, jak se k němu dostat (např URN: ISBN-80-1798-…)
- Schéma
    - je položka, kterou se definuje typ zdroje a přístupová metoda ke zdroji. http, ftp, mailto:, file:

URI schema http

- http://pocitac/adresare/soubor
- Dále je možné připojit
    - ?parametr=hodnota;parametr=hodnota
    - #vnitřní odkaz

HTTP (Interakce – 2. pilíř)

http = Hypertext Transfer Protocol

= protokol určený původně pro výměnu hypertextových dokumentů ve formátu HTML obvykle TCP/80

Základní model protokolu:

1. navázaní spojení
2. zaslání požadavku klientem
3. zaslání odpovědi serverem
4. ukončení spojení

Struktura odpovědí a požadavků HTTP

- **Struktura požadavku v http 1.0 a 1.1**

```
Metoda URL_dokumentuverze_HTTP
Hlavičky
Prázdná řádka
Tělo požadavku
```

- **Struktura odpovědi v http 1.0 a 1.1**

```
Protokol stavový_kódstavové_hlášení
Hlavičky
Prázdná_řádka
Obsah_odpovědi
```

Metody HTTP

- **OPTIONS** (od 1.1), Dotaz na server, jaké podporuje metody.
- **GET** (od 0.9), Požadavek na uvedený objekt se zasláním případných dat (proměnné prohlížeče, session id, …). Výchozí metoda při požadavku na zobrazení hypertextových stránek, RSS feedů aj. Celkově nejpoužívanější.
- **HEAD** (od 1.0), To samé jako metoda GET, ale už nepředává data. Poskytne pouze metadata o požadovaném cíli (velikost, typ, datum změny, …).
- **POST** (od 1.0), Odesílá uživatelská data na server. Používá se například při odesílání formuláře na webu. S předaným objektem se pak zachází podobně jako při metodě GET. Data může odesílat i metoda GET, metoda POST se ale používá pro příliš velká data (víc než 512 bajtů, což je velikost požadavku GET) nebo pokud není vhodné přenášená data zobrazit jako součást URL (data předávaná metodou POST jsou obsažena v HTTP požadavku).
- **PUT** (od 1.1), Nahraje data na server. Objekt je jméno vytvářeného souboru. Používá se velmi zřídka, pro nahrávání dat na server se běžně používá FTP nebo SCP/SSH.
- **DELETE** (od 1.1), Smaže uvedený objekt ze serveru. Jsou na to potřeba jistá oprávnění stejně jako u metody PUT.
- **TRACE** (od 1.1), Odešle kopii obdrženého požadavku zpět odesílateli, takže klient může zjistit, co na požadavku mění nebo přidávají servery, kterými požadavek prochází.
- **CONNECT** (od 1.1), Spojí se s uvedeným objektem před uvedený port. Používá se při průchodu skrze proxy pro ustanovení kanálu SSL.

Stavové kódy:

- 1xx**: informativní kód** (100 – Continue, 101 – Switchprotocol)
- 2xx**: úspěšné vyřízení požadavku** (200 – OK, 201 – Created, 202 –Accepted)
- 3xx**: přesměrování** (301 – Movedpermanently)
- 4xx**: chyba klienta** (400 – Badrequest, 401 –Unauthorised)
- 5xx**: chyba na straně serveru** (500 – Internal server error)

Hlavičky HTTP

Hlavičky jsou rozděleny do tří skupin:

- Obecné hlavičky poskytující univerzální informace o zprávě

- Hlavičky dotazu/odpovědi, které popisují dotaz/odpověď
- Hlavičky těla, které popisují tělo zprávy

Ukázky hlaviček, podle toho k čemu se dají použít:

Nejdůležitější hlavičky – vybrané příklady

- **Date:** datum a čas požadavku/odpovědi
- **Content-Type:** druh zasílaných dat
- **Host:** doménová adresa serveru – umožňuje správnou funkci více virtuálních serverů na jedné společné adrese
- **Location:** přesměrování na jinou stránku

Ovládání vyrovnávacích pamětí, proxy serverů a načítání stránek

- **Cache-Control:** řízení proxy serverů a vyrovnávacích pamětí
- **Pragma:** vyhrazeno pro nestandardní informace (nejčastěji zákaz kešování pro starší prohlížeče)
- **Expires:** datum, kdy vyprší platnost stránky
- **If-Modified-Since:** podmíněné načtení stránky
- **Last-Modified:** datum poslední modifikace souboru

Domlouvání obsahu

- **Accept:** seznam typů dat podporovaných klientem
- **Accept-Charset:** seznam kódování, které podporuje klient
- **Accept-Language:** seznam podporovaných jazyků
- **Allow:** seznam metod, kterými je dostupný určitý objekt

Identifikační údaje

- **User-Agent:** identifikace klienta
- **Server:** identifikace serveru
- **Referer:** adresa stránky, kde bylo získáno URL právě kladeného požadavku (lze použít pro analýzu typu „odkud přišli“)

Cache

= dočasně uložená pomocná data

V kontextu webu se vyskytují dva druhy:

- **cache na serverech** cestou (také proxyny)
- **cache na klientovi** (na disku uživatele a v prohlížeči).

Cache pomáhají **zrychlit internet**, protože si pamatují, co bylo staženo. Každý objekt (stránka, obrázek) má nějak nastaveno, kdy se z cache paměti může smazat a kdy se má začít dívat, jestli nebyl objekt změněn. To určují http hlavičky expires, cache-control a pragma. Nastavují se konfigurací serveru. Prohlížečové cache (ne serverové, proxyny) lze ovlivnit též pomocí meta tagů http-equiv.

Řízení cashe

Cache musí odpovědět poslední odpověď drženou cache, která odpovídá dotazu a platí jedna z podmínek.

- Odpověď byla revalidována na serveru.
- Odpověď je dostatečně čerstvá
- Odpoví hláškou 303, 304 nebo chybou.

Expirační pravidla

- Dobu platnosti, a nutnost revalidace řídí server

- Používá Expire hlavičku nebo max-age
- Pokud neuvádí cache používá heuristické algoritmy.

Direktivy pro řízení cache

- Omezení, co je pro cache – nastolí server:
    - Public, private, no-cache
- Omezení, co může být vloženo – klient i server: no-store
- Modifikace expiračního mechanismu – klient i server:
    - max-age, min-fresh, max-stále
- Ovládání revalidace a aktuallizace

Bezpečnost na WWW

HTTP nedává žádnou možnost zabezpečení komunikace.

Firma Netscape vytvořila protokol SSL pro přenos šifrovaných dat.

SSL

SecureSocketsLayer, SSL (doslova vrstva bezpečných socketů) je protokol, resp. vrstva vložená mezi vrstvu transportní (např. TCP/IP) a aplikační (např. HTTP), která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran. Následovníkem SSL je protokol Transport LayerSecurity (TLS).

Protokol SSL se nejčastěji využívá pro bezpečnou komunikaci s internetovými servery pomocí HTTPS, což je zabezpečená verze protokolu HTTP. Po vytvoření SSL spojení (session) je komunikace mezi serverem a klientem šifrovaná a tedy zabezpečená.

6. Základy webových informačních technologií

Původně byl web navržen pro sdílení statických dokumentů
Dnes v něm lze používat různé aplikace – trendem je posun webu k nové platformě pro vývoj aplikací
Výhody: velký rozsah, nemusí se instalovat u klienta
Nevýhody: programátor musí znát několik technologií (HTTP, HTML, CSS, PHP, JavaScript, …)

Přístupy k tvorbě dynamických WWW

Kosek uvádí tyto přístupy:

- výkonná část aplikace běží přímo v prohlížeči (JavaScript, Visual Basic Script,…)
- výkonná část aplikace běží na serveru (PHP, ASP, CGI,…)
- nejčastěji se používá kombinace obou technologií

Na straně klienta

Výhody použití rozšiřujících klientských technologií

- Stránky jsou interaktivní
- Rychlejší odezva
    - Např. kontrola dat z formulářů přes JavaScript – nemusí se odesílat data na server, zkontroluje si je sám klient
    - Proto nekomunikuje vždy se serverem
    - Nemusí se přenášet a překreslovat celá stránka
    - Dochází k odlehčení zátěže serveru

Nevýhody:

- **Hlavně:** na klientské technologie se nelze spoléhat na 100% (proto je dobré je použít jako pomocné a důležitou, citlivou práci provádět na straně serveru)
- V důsledku principů fungování http protokolu nemůže serverová část spoléhat na jakékoliv předzpracování, které prováděl klient
- Uživatel je může vypnout (zakázat)
- Podpora technologií v jednotlivých prohlížečích není dokonalá
- Některé technologie mohou neoprávněně manipulovat s daty v počítači klienta

Skripty

- Přímo do HTML stránky lze vložit jednoduché programy (scripty) nebo lze htl stránce říct, aby načitala externí část kódu
- Poté reagují na události vyvolané uživatelem
- Program může manipulovat s dokumentem a prohlížečem
- Teoreticky lze použít libovolný jazyk
    - nejpodporovanější je **JavaScript**
    - některé prohlížeče podporují VBScript, Python

Příklad JavaScriptu

- Reaguje na nějakou definovanou událost (každý element má seznam podporovaných atributů, ty odpovídají jednotlivým událostem)
- Často je vložen přímo do HTML (ale může být jen nalinkován)
- Jako hodnota atributu se uvádí kód, který se má provést
- **<tag událost="obslužný kód">…</tag>**
- Pomocí událostí lze vytvářet interaktivní stránky – ty pak reagují na události (uživatelské události, události okna a dokumentu, události myši a události klávesnice)

Na straně serveru

- na serveru je dynamicky generováno HTML na základě požadavku uživatele
- do prohlížeče je odesílán již jen čistý HTML kód
- není potřeba žádný speciální prohlížeč, lze použít libovolný se základní podporou HTML
- v případě potřeby lze na serverem generovaných stránkách použít i klientské technologie (např. JavaScript)

Server Side Includes (SSI)

- Nejstarší a nejrozšířenější druh serverem vkládaných vsuvek
    - Dnes jej podporují všechny dostupné www servery
- Do HTML jednoduché instrukce, které provádí přímo webový server
- Syntaxe: <!--#příkazparametry-->

CGI (Commn Gateway Interface)

- CGI script je externí program, který je na požadavek od uživatele spuštěný WWW serverem jako samostatný proces.
- CGI skript je program, který používá rozhraní CGI

PHP (Personal Home Pages)

- přímo do HTML kódu se zapisují jednoduché příkazy
- jednoduchá syntaxe založená na C, Perlu a Javě
- speciálně navržený jazyk pro tvorbu webových aplikací
- velmi rozsáhlá knihovna funkcí
- nezávislost na platformě – může spolupracovat s v podstatě libovolným serverem na libovolné platformě

- OSS – dostupný zdarma včetně zdrojových kódů
- Často se setkáme s: kombinace Linux, Apache, MySQL a PHP – je to zdarma

Java Server Pages (JSP)

- založený na Javě, podobný ASP a PHP
- mají silnou typovou kontrolu
- kompilace stránek do servetů – Javových tříd
- může komunikovat s jinými třídami Javy – oddělení designu od logiky
- podpora sessions

Active Server Pages (ASP)

- přímo do HTML kódu se zapisují jednoduché příkazy
- ASP je jen jakýsi framework
    - lze použít libovolný jazyk podporující Active Scripting
        - standardně JScript a VBScript
        - třetí firmy dodávají Perl, REXX, Python
- ve všech jazycích jsou dostupné základní objekty s důležitými informacemi (data z formulářů apod.)
- standardní součást webových serverů MS
- podpora jiných serverů a platforem je velice slabá

ASP.NET

ASP.NET je součást .NET Frameworku pro tvorbu webových aplikací a služeb. Je nástupcem technologie ASP (Active Server Pages) a přímým konkurentem JSP (Java Server Pages).

- s klasickými ASP nemá skoro nic společného
- vyvíjí se jako klasická klientská aplikace – prvky uživatelského rozhraní a obsluha událostí
- ASP.NET si webový server přeloží do nativního kódu, který se stará o postupné zasílání HTML kódu a obsluhu formulářových dat
- vygenerovaný kód detekuje použitý prohlížeč a tomu přizpůsobí generovaný HTML a JavaScriptový kód
- VisualStudio.NET umožňuje aplikace vyvinout pouhým „naklikáním“
- později byly pro ASP.NET vytvořeny další nastavby – např. ASP.NET MVC nebo Razor

Vytváření relace a udržování informace o relaci

Omezení HTTP

- protokol HTTP je **bezstavový**
- server nemá stále spojení s klienty a nemůže je proto jednoznačně identifikovat
- velké komplikace pro webové aplikace, které vyžadují stavovou informaci – např. nákupní košík, přihlášení atp.
- Jde o to, že protokol HTTP je bezstavový, tudíž každý klientský požadavek je zcela samostatný a nijak nesouvisí s tím předchozím. Což je v mnoha případech velmi nepříjemné (např. autentifikace). Webovou aplikaci najednou obvykle používá více uživatelů. Aplikace musí nějak poznat, který klient požadavky posílá, a musí si nějak uchovávat, v jakém je webová aplikace pro tohoto konkrétního uživatele stavu, co již tento uživatel během relace provedl, jestli je přihlášen, jaké zboží dal do košíku atd..

Řešení

Přenášení údajů v URL a skrytých polí formuláře

- nebezpečné – všechny stavové informace jsou v každém požadavku/odpovědi
- zbytečně zvyšuje přenosovou kapacitu

- velmi pracné na implementaci – za každý odkaz a do každého formuláře se musí přidat všechny stavové proměnné

Cookie

- krátká informace, kterou si server uloží v prohlížeči
- při následujících přístupech k témuž serveru je cookie zaslána zpět
- cookie je vázána na server a případně i na adresář – informace se nedostanou k tomu, komu nepatří
- časová platnost cookie
- session cookie – platí do té doby, než se vypne prohlížeč
- nastavena na konkrétní délku
- nebezpečné – všechny stavové informace jsou v každém požadavku/odpovědi
- implementace je velice snadná
- některé starší prohlížeče cookies nepodporují, novější umožňují cookies vypnout
- na cookies bohužel nelze spoléhat

Session proměnné

- Každému uživateli server přiřadí unikátní identifikátor (tzv. session-id)
    - předává se s každým požadavkem pomocí cookie nebo parametrů v URL, resp. skrytých polí ve formuláři
    - session-id je konstruováno tak, aby bylo těžko odhadnutelné (většinou náhodné číslo + hashovací funkce MD5 nebo SHA)
    - pro každé session-id má webový server vyhrazen prostor pro ukládání dat (proměnných)
- session je poměrně bezpečné předává se jen session id
- malá kapacita na síť
- snadná implementace (v PHP pomocí proměnné $_SESSION[„název“])

Web Storage

- úložiště dat na klientovi (neřeší problém identifikace klienta na serveru)
- součást HTML5, podporováno všemi moderními prohlížeči
- pojme více dat než cookies a nepřenáší se na server, data zůstávají u klienta
    - localStorage – je persistentní i přes uzavření prohlížeče
    - sessionStorage – platné jen po dobu jedné relace

Binární soubory a rozhraní serveru

Interakce s binárními programy:

- CGI (Common Gateway Interface)
- FAST-CGI
- Apache API – možnost tvorby vlastních modulů pro webový server Apache v jazyce C
- Další aplikační vrstvy

CGI (Common Gateway Interface)

- Původní model interakce s aplikací (definuje způsob komunikace web-serveru a aplikací)
    - Dnes se moc nepoužívá, nahradily ho skriptovací jazyky (např. PHP)
- CGI skript je program, který používá rozhraní CGI
- CGI skripty lze psát v téměř libovolném jazyce, stačí dodržet konvence rozhraní CGI
    - shell, Perl, C/C++, Pascal, Python, …
- Server uloží parametry do „environment variable“ a zavolá aplikaci
- Pro POST příkaz server dá do environment variable CONTENT_LENGTH délku dat a na STDIN pošle data.
- Server vrátí zcela naformátovanou HTML stránku na STDOUT

FastCGI

- vylepšená varianta rozhraní CGI, snižuje zátěž serveru
- každý skript se do paměti načítá jen jednou, pak postupně obsluhuje další požadavky
- web-server s aplikací komunikuje pomocí TCP/IP
    - web-server a aplikaci je možné rozdělit na samostatné počítače
    - primitivní řešení load-balancingu

HTML

= jazyk pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu Dnes se používá ve verzi HTML 5.0. V návrhu jsou další dvě verze 5.1 a 5.2

Syntaxe

- Jednotlivé části HTML stránky se označují pomocí elementů
- Každý element se skládá z počátečního tagu a ukončovacího tagu (některé elementy nemají ukončovací tagy
, <hr>)
- Některé elementy mají atributy , které upřesňují jeho význam

Historie

- HTML 2.0 – první formální specifikace, základní formátování a strukturování dokumentu, obrázky, formuláře
- HTML 3.0 – (r.1995) moc ambiciózní (matematické vzorečky atp.) žádná firma nebyla schopna naprogramovat jeho podporu ve svých produktech – nebylo přijato jako standard
- HTML 3.2 – (r.1996) něco se chytlo, tak W3C vybralo podmnožinu a tu vydalo jako standard
- HTML 4.0 – (r.1997)
    - Velký skok vpřed – podpora CSS, skripty vložené do stránky, multimediální objekty, rozšíření formulářů
- HTML 4.01 – (r.1999) – oprava drobných chyb

7. Principy a aplikace objektového paradigma

Objekty

Jedná se o abstrakci z reality, každý objekt představuje spojení dat (údajů, proměnných, datových atributů) a činností s těmito daty (metod). Tato abstrakce je vždy účelová, o každém reálném objektu se sledují ty údaje, které jsou relevantní pro aplikaci. Pokud chceme pracovat s objekty, je nutné vědět, jaké jsou obecné vlastnosti objektů.

Obecné objektové vlastnosti:

- používání abstrakce
- definování tříd objektů
- existence objektů (instancí)
- komunikace objektů (posílání zpráv, volání metod)
- zapouzdření a ukrývání implementace
- dědičnost
- polymorfismus

Abstrakce

Abstrakce je základní objektovou vlastností. Skutečnost, kterou chceme do programu promítnout, musíme vždy zjednodušit, pracovat jen s těmi daty, která jsou pro nás důležitá.

První příklad:

Když chceme udělat počítačovou evidenci knih, které jsou k dispozici v obchodě, bude základem abstrakce knihy. V knihkupectví nás bude pravděpodobně u každé knihy zajímat: autor, název, ISBN, vydavatel, žánr, cena, počet kusů na skladě. S knihou budeme provádět např. tyto činnosti: založení nové knihy, změna množství na skladě, změna ceny.

Druhý příklad:

Chceme napsat aplikaci pro kreslení. Tvary (čtverce, obdélníky, kruhy, trojúhelníky atd.), které budou nakresleny, jsou objekty. U každého nakresleného tvaru musíme sledovat např. tato data: souřadnice umístění, rozměry tvaru, barvu čáry, barvu výplně. Metody neboli činnosti, které bude možno v takové aplikaci provádět s jednotlivými tvary, budou například tyto: nakreslení, zvětšení, zmenšení, změna barvy, posun, vymazání.

Třída a instance

Třída je obecný popis, ve kterém se deklarují (určí) data (datové atributy), která budou popisovat stav objektu, a metody, které definují činnosti, jaké je možné s objekty provádět. Programátor definuje, které údaje se budou o objektech sledovat a jaké činnosti bude s těmi daty možno provádět a jak se to bude dělat.

V programu se poté vytvoří několik instancí této třídy. **Instance** je tedy vytvořena v paměti počítače a vytváří jakýsi obraz reálného objektu (např. účet Josefa Nováka).

Data, která budeme o každé instanci sledovat, označujeme jako **datové atributy instance**. Činnosti, které je možné s danými instancemi provádět, označujeme jako **metody (metody instance).**

Volání metod (posílání zpráv)

Každá aplikace je tvořena několika třídami, v rámci běhu aplikace jsou vytvářeny instance těchto tříd a volány jejich metody. Volání metod se podobá posílání SMS z mobilního telefonu. Můžeme poslat SMS jen tomu, na koho máme číslo.

Ucet mujUcet; - identifikátor použitý na uložení odkazu musí být typu Ucet

mujUcet = new Ucet (1,”Pepa”); - vytvoření nové instance třídy

Volání metod ze třídy Ucet:

mujUcet.vloz(100);

double stavMehoUctu = mujUcet.getStav();

V případě volání metod stejné instance používáme klíčové slovo **this**.

Zapouzdření

Zapouzření (encapsulation) popisuje princip umisťování dat a souvisejících metod k sobě – do jednoho objektu, do jedné metody atd. Zapouzdření musí být podporováno vhodnou jazykovou konstrukcí, v Javě i ostatních objektových jazycích se realizuje pomocí tříd a vytváření instancí.

Při zapouzdřování objektů je vhodné **dodržovat tato pravidla**:

- Snažit se umístit data a operace pracující s daty (metody) do stejné třídy.
- Třída by neměla obsahovat jen část dat či část metod, ani by neměla obsahovat více dat či metod, než je nutné pro činnost, za kterou třída odpovídá. Častější chybou je, že třída obsahuje více dat než kolik potřebuje.
- Jednotlivé zprávy posílané instanci by pokud možno měly být na sobě nezávislé, metody by měly požadovat jen nezbytně nutné parametry.

Ukrývání implementace

Možnost používat metody objektů bez znalosti jejich implementace se nazývá ukrývání implementace. Každý objekt poskytuje svému okolí metody, které je možné zavolat. Seznam těchto metod (a dostupných datových atributů) je

označován jako **veřejné rozhraní třídy**. V tomto rozhraní by neměly být zahrnuty datové atributy, ty by měly být schovány uvnitř instance a měly by být přístupné pouze pomocí metod.

Předpokladem pro ukrývání implementace je, že programovací jazyk podporuje zapouzdření. Oba pojmy spolu úzce souvisí, což někdy vede k jejich nepřesnému používání či vzájemnému zaměňování.

Následující modifikátory uvádějí možnost přístupu k datovému atributu nebo metodě:

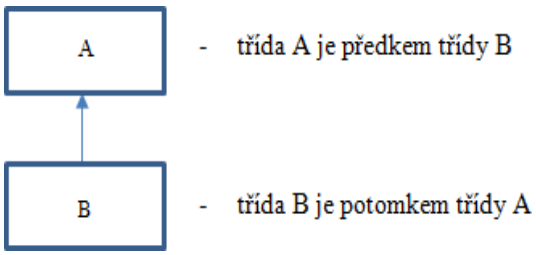
- private
- (nic neuvedeno)
- protected
- public

Označíme-li nějaký datový atribut nebo metodu jako **private**, znamená to, že je přístupná pouze z metod instance. Jako druhý modifikátor vlastně není nic uvedeno, ale znamená to, že pokud neuvedeme žádný modifikátor přístupu, použije se **přátelský přístup**. Datové atributy a metody jsou v tomto případě přístupné v rámci balíčku. Datové atributy a metody, které mají označení přístupu **protected**, jsou přístupné v rámci balíčku a také z potomků v rámci dědičné hierarchie. Označení přístupu **public** znamená, že daný datový atribut nebo metoda jsou přístupné z jakékoliv jiné třídy.

Dědičnost

Dědičnost je jednou z forem **znovupoužitelnosti** – vytvářená třída (potomek) do sebe absorbuje datové atributy a dědí metody z jiné třídy (předek) a dále je rozšiřuje a upravuje.

V diagramu tříd se dědičnost vyznačuje pomocí šipky, u které trojúhelník na konci směřuje k předkovi.



Dědičnost není pouze jednoúrovňová – potomek nějaké třídy může mít dále své potomky. Tito potomci dědí metody od všech tříd na vyšší úrovni. Takto vzniká hierarchie tříd, ve které není omezen počet úrovní.

Java podporuje jednonásobnou dědičnost – každá třída může a musí mít právě jednoho předka. Stromová hierarchie tříd začíná třídou Object (tato jediná třída nemá předka), všechny třídy jsou přímým či nepřímým potomkem této třídy.

Dědičnost by se měla používat v situacích, kdy potomek je podtypem svého předka, tj. existuje mezi nimi vztah. Pokud má být nějaká třída B potomkem třídy A, měli bychom si kladně odpovědět na tyto otázky: „B je A?“ či „Je každý B také A?“.

Nemůžeme-li na takou otázku odpovědět kladně, neměli bychom používat dědičnost. Jsou vztahy, které lze vyjádřit pomocí „je částí“ („part-of“) a „má“ („has-a“). Pokud jsou mezi třídami tyto vztahy, nepoužívá se dědičnost, ale většinou kompozice. Důležité je též si uvědomit, že dědičnost vyjadřuje vztah tříd, ne vztah instancí. Pro objektový jazyk C++ s vícenásobnou dědičností se obvykle doporučuje, aby základem nějaké dědičné hierarchie byla abstraktní třída. Toto neplatí plně pro jazyky s rozhraními, jako je např. Java, kde se jako základ dědičné hierarchie obvykle používá rozhraní.

Důvody použití dědičnosti

Dědičnost se využívá v různých situacích, za různými účely. V praxi lze důvody pro použití dědičnosti obtížně odlišit.

1. **Specializace**

Jedním z nejčastějších důvodů pro použití dědičnosti je specializace existujících tříd a objektů. Při specializaci získává třída nové datové atributy a chování proti původní třídě.

Příklad: specializace s bankovním účtem a žirovým účtem

2. **Překrývání metod a polymorfismus**

Častým důvodem k dědičnosti je možnost využití překrývání metod a následně polymorfismu – různí potomci mají rozdílně implementována některá chování (některé metody). Při volání takové metody programátor nemusí uvažovat o tom, které konkrétní instanci posílá zprávu, neboť každá instance má k sobě přiřazen svůj specifický kód.

3. **Znovupoužití kódu**

Jednou z prvních motivací pro dědičnost bylo umožnit nové třídě znovu použít kód, který již existoval v jiné třídě. Pokud vede k dědičnosti pouze tento motiv, vznikne hierarchie, kdy věcně nelze přetypovat potomka na předka.

Polymorfismus

Pojem pochází z řečtiny a znamená mnohotvarost. V OOP vyjadřuje situaci, kdy při stejném volání provádí různý kód. Která konkrétní metoda se provede, závisí **na předávaných parametrech a objektu**, kterému je zpráva předána.

- přetěžování metod (overloading), též ad-hoc polymorphism (metoda má více než jednu definici ve stejném „scope“ (stejné jméno, různé parametry))
- překrývání metod (overriding), též subtype polymorphism (u potomka je stejná deklarace metody, ale jiná implementace)
- parametrický polymorfismus – např. šablony v C++, nemá pevně stanovený typ argumentu

8. Vývoj řízený testy

= **TDD – Test Driven Development**

Myšlenka: **definovat nejprve sadu testů a teprve pak psát testovaný program**. Při psaní programu se pak stačí soustředit pouze na to, aby výsledný program prošel napsanými testy. Jakmile program těmito testy projde, programátor se zamyslí, jak by jej měl dále vylepšit, zase napíše testy a opět se snaží upravit program tak, aby testy prošel.

Refaktorování je nedílnou součástí vývoje řízeného testy – jedná se o proces provádění změn v softwarovém systému takovým způsobem, že nemají vliv na vnější chování kódu, ale vylepšují jeho vnitřní strukturu. Je to disciplinovaný způsob pročišťování kódu s minimálním rizikem vnášení chyb. Zlepšování návrhu kódu poté, co byl napsán.

Výhody TDD:

- Při psaní programu může programátor kdykoliv spustit předpřipravenou sadu testů a průběžně na nich kontrolovat, kolik jich už proběhlo a kolik práce mu ještě zbývá.
- Ve chvíli, kdy proběhnou všechny testy, je programátor s prací hotov a nemusí již přemýšlet nad tím, jestli v jeho programu nejsou chyby.

„Žádný software není bezchybný“

- Čím později je chyba objevena, tím dražší je její odstranění
- Náklady na odstranění chyb v průběhu životního cyklu projektu stoupají 100x až 1000x

Testování Softwaru

- Je provádění funkcí programu v definovaném okolí a porovnání dosažených výsledků s těmi očekávanými, aby bylo zjištěno, jak dalece se program chová, tak, jak vyžaduje jeho specifikace.
- Cílem testování softwaru je nalézt chyby ve zkoumaném softwaru, a tak vytvořit předpoklady pro jejich odstranění, což vede ke zvýšení kvality softwaru.
- Dalším cílem je dosažení spolehlivosti.
- Testy, i rozsáhlé, **nemohou zpravidla zaručit bezchybnost softwaru**.

- **Oblast použití:** software, jehož kvalita má být dokázána.
- Jen výjimečně postačují samotné testy k důkazu bezpečnosti.
- **Soudobý trend – Vývoj řízený testy (TDD – Test Driven Development)**

Definice:

- Proces získání důvěry v to, že program nebo systém dělá, co se od něj očekává. (Hetzel 1973)
- Provozování programu nebo systému za účelem hledání chyb. (Myers 1979)
- Jakákoliv aktivita zaměřená na vyhodnocení vlastností a schopností programu nebo systému a určení, zda odpovídají očekávaným výsledkům. (Hetzel 1983)

Testování – fakta

- testování pokrývá 50% času projektu
- v tomto je zahrnuto i ladění a všechny typy testování,
- vývojářské testování – 10 až 25% času projektu,
- 80% chyb je ve 20 % kódu,
- většinu chyb (> 95%) vytvoří programátoři aplikace, tj. nejsou v OS, databázi, knihovnách, ...
- třetina chyb jsou obyčejné překlepy a pravopisné chyby.

Testování jako projekt

K testování SW je vhodné přistoupit jako k jakémukoliv jinému projektu. Celý projekt lze rozdělit do 4 hlavních částí:

- **Příprava na testování** – v této fázi vznikají obvykle tyto dokumenty: testovací plán, testovací scénáře a testovací případy.
- **Provedení testů** – v souladu s testovacím plánem pak probíhají vlastní testy podle testovacích scénářů.
- **Vyhodnocení testů** – pokud by se vyhodnocování neprovádělo, tak by testování ani nemělo smysl.
- **Rozhodnutí o dalším postupu** – zopakování některých testů apod.

Axiomy testování

- Žádný program nelze otestovat komplexně (množství různých vstupů do aplikace).
- Testování je postaveno na riziku (co když bude chyba zrovna v tom, co neotestujeme).
- Testování nikdy neprokáže, že chyby neexistují.
- Čím víc chyb najdete, tím víc jich tam je.
- Ne všechny chyby se odstraní.
- Cíl testování je v rozporu s cíli ostatních aktivit – zde je cílem nalézt chybu.

Místo testování při řízení projektu

- pokud testujeme v počátečních fázích vývoje, snižuje testování náklady na pozdější odstraňování chy.
- Důležitým faktorem ovlivňujícím cenu je část životního cyklu, v jaké se software při objevení problému nachází. Čím později je chyba odhalena, tím větší množství peněz stojí její odstranění.

Kategorizace testů

- z hlediska předmětu – rozlišujeme úrovně testů
- z hlediska cíle – rozlišujeme typy testů

Z hlediska předmětu – Úrovně testů

- **Jednotkové testování** – testování nejmenších vymezených částí systému (jednotek), provádějí vývojáři
- **Integrační testování** – testování zaměřené na vzájemné vazby a propojení částí systému, ověřuje, že jednotky dodané vývojovým týmem lze integrovat, tj. vytvořit funkční build.

- **Smoke test** – ověřuje stabilitu buildu, provádí se před systémovým testem – eliminuje situaci, kdy po zapojení většího počtu testerů systém selže a není možné v testech pokračovat.
- **Systémové testování** – testování systému jako celku, od vstupu po výstup. Systémový test představuje hloubkový test systému, při kterém se zpravidla provádějí téměř všechny typy testů.
- **Akceptační testování** – testování za účelem prokázání splnění definovaných akceptačních kritérií, testy provádí koncový uživatel na vlastním testovacím prostředí, tím je zaručena reálnost testovaných situací.

Z hlediska cíle – Typy testů

podle základních atributů kvality FURPS (/FURPS+)

- **Functionality** = testy funkčnosti (testování funkčních požadavků)
- **Usability** = testy použitelnosti (test uživatelského rozhraní, nápovědy, školících materiálů, dokumentace)
- **Reliability** = testy spolehlivosti (frekvence selhání, možnosti obnovitelnosti, nestandartní podmínky)
- **Performance** = testy výkonu (časová odezva, propustnost, přesnost, dostupnost)
- **Supportability** = testy podpory (adaptabilita, udržovatelnost, internacionalizace, konfigurovatelnost)

Další dělení testů

Podle způsobu provedení

- **Manuální** – testy provádí člověk (obvykle tester) podle předem stanovených testovacích směrnic, vlastních zkušeností, atd.
- **Automatické** – spouštění předem naprogramovaných testů ve speciálním programu.

Podle potřeby spouštět program

- **Statické** – provádí se bez nutnosti spuštění programového kódu. Testování probíhá analýzou kódu (code review). Analýza může být provedena buď ručně, nebo pomocí speciálních nástrojů.
- **Dynamické** – provádí se spouštěním programu a kontrolou jeho funkčnosti. Testování probíhá prováděním ručních testů, nebo spouštěním předem připravených automatických testů.

Funkční/nefunkční/testy spojené se změnami

- **Funkční testy** – založené na funkcích, vlastnostech a možnostech interakce programu s jinými systémy; např. testy funkcionality, bezpečnostní testy a další.
- **Nefunkční testy** – komplexní testování toho, jak systém funguje; např. zátěžové, stresové testy, testy stability, použitelnosti atd.
- **Testy spojené se změnami** – po provádění nezbytných změn, například oprav chyb nebo defektů, musí být software znovu otestován, aby se zjistilo, zda došlo k odstranění problému; např. regresní testování.

Podle znalosti kódu

- **Testování „černé skříňky“ (black box testing)** – tester má přístup k programu pouze přes stejné rozhraní jako zákazník nebo uživatel. Zadává vstupy, postupuje připravených scénářů a na konci porovnává výsledek, zda se shoduje s očekáváním.
- **Testování „bílé skříňky“ (white box testing)** – tester má přístup ke zdrojovému kódu programu a zná vnitřní datové a programové struktury. Při testování tuto znalost využívá.
- **Testování „šedé skříňky“ (grey box testing)** – tester má přístup ke zdrojovému kódu, ale obvykle při spouštění testu nepotřebuje ke kódu přistupovat.

Regresní testování

- Takový test systému, při kterém jsou kromě nově přidaných funkčností **testovány i všechny funkčnosti z dřívějších buildů**.

- Cíl: zabránit regresním chybám (rozbije se to, co dříve fungovalo; případně navrácení již odstraněných chyb).

Závislé x nezávislé testování

- **Závislé testování** provádějí role, které mají jinou specializaci než testování (testují produkty vlastní práce), příklad vývojářské testování.
- **Nezávislé testování** provádějí role, které testují výsledek práce jiných rolí, například testeři, zástupci zákazníka podílející se například na akceptačním testování nebo beta testování.

JUnit testování

Open-source nástroj (framework) pro testování tříd napsaných v Javě

- Testovací rámec pro tvorbu a spouštění automatizovaných jednotkových testů
- Pro každou třídu se vytvoří soubor testů, které kontrolují, zda jednotlivé veřejné metody vracejí k zadaným vstupům očekávané hodnoty.
- Použití např. v metodice TDD (Test-Driven Development) – testy píšeme před zápisem samotného kódu
- TestCase = nejčastěji užívaná třída frameworku JUnit – za pomoci dědičnosti jsou z ní vytvářeny odvozené třídy, které obsahují samotný testovací kód.

Každý z testů ve třídách odvozených od TestCase má **3 části (metody**):

- **setUp()**
    - metoda, která vytváří tzv. „přípravek“ = sada objektů, které se vytvoří před spuštěním každého testu,

doplněná případně o nastavení nějakých okrajových podmínek. V této metodě je realizována tvorba a uložení instancí do atributů, nastavení vstupních hodnot atd.

- **testNazevTestu()**
    - tato metoda zkoumá a vyhodnocuje chování testovaných prvků. Jméno testovací metody musí vždy začínat prefixem test. Framework díky tomuto předpokladu automaticky pozná, že se jedná o testovací metodu.
- **tearDown()**
    - = úklid po testu – uvolnění zdrojů

Uvnitř testovací metody se nachází tzv. **potvrzovací metody** (nazývány také **testovací konstrukce**):

- v testovacích metodách se používají pro ověřování, zda je předpokládaný výsledek rovný výsledku skutečnému
- Základními potvrzovacími metodami jsou metody assertEquals(x1,x2), kterých je celá řada a liší se pouze tím, jakého typu jsou jejich parametry. Těmto metodám předáme v prvním parametru očekávanou hodnotu a ve druhém hodnotu, ke které dospěl program. Pokud se tyto hodnoty liší, metody vypíší v testovacím okně očekávanou a obdrženou hodnotu a daný test ukončí.
- Dále např.: assertTrue, assertNull, fail a další.

9. Prvky funkcionálního programování aplikované např. v Jave (lambda výrazy a streamy)

Funkcionální paradigma je založeno na zápisu programu ve **tvaru výrazu namísto příkazů**. Nejdůležitějšími složkami těchto výrazů **jsou funkce**, které lze předat jako argument jiné funkce. Výraz se používá, **aby vytvořil hodnotu**, zatímco příkaz se používá pro přiřazení hodnoty. Funkcionální (deklarativní) programování se vyznačuje hlavně tím, že se nezabývá postupným vypracováním programu krok po kroku, to znamená, že nemusí psát dlouhé kódy, kde je popsán celý postup, jak se má zadaný program řešit – to ho v podstatě vůbec nezajímá. To je ponecháno různým nástrojům, které prostředí jazyka poskytují. Díky nim pro nás řada pohledů zůstává skryta. Zatímco deklarativní zápis programu představuje již samotné řešení problému zapsané v jistém tvaru.

Výpočet funkcionálního programu spočívá ve **zjednodušování výrazu** až do doby, kdy výraz dále zjednodušit nelze. Běh funkcionálního programu je založen na principu **skládání funkcí**. Skládání znamená, že můžeme spojit více funkcí dohromady, tedy argumentem funkce může být nějaká jiná funkce.

V tomto druhu programování se hodně vyskytuje **tzv. rekurze**. Jde vlastně o funkci (metodu), která volá sama sebe. Díky rekurzi se ve funkcionálním programování nemusí tak využívat cykly (for, do-while, while). Pokud se jedná o proměnné, tak ani ty se zde nevyužívají tak často. Bude-li se chtít v jisté části programu použít nějaká hodnota, tak namísto použití proměnné, se použije naimplementovaná metoda se správným návratovým typem. Čím se hlavně vyznačuje funkcionální programování, jsou tzv. lambda výrazy. Ty umožňují předávat nějakou část zdrojového kódu podobně jako proměnné. Část kódu, která by zabrala i několik řádků, se tak dokonce může **vejít pouze na jednu**.

Funkcionální programování se vyznačuje několika charakteristikami

- **Funkce první třídy, funkce vyššího řádu – takové** funkce, které lze bez starostí předávat jako argumenty funkcí, výsledky funkcí nebo je lze ukládat do proměnných
- **Neměnná data – Jde** například o objekt, jenž nelze po vytvoření již změnit
- **Čisté funkce – se** vyznačuje tím, že nemění hodnoty zadávaných parametrů a vrací hodnoty pomocí příkazu return.
- **Rekurze – O** rekurzi lze mluvit tehdy, když nějaká funkce ve svém těle volá buď sama sebe, nebo nějakou jinou metodu, čímž dochází k opakování programu, a tak nahrazuje cykly
- **Manipulace se seznamy**
- **Odložené vyhodnocování – vyhodnocení** nějakého výrazu je odloženo až do okamžiku, kdy je to opravdu potřebné.

Funkční rozhraní

Jako funkční se označuje takové rozhraní, které **obsahuje hlavičku** jediné abstraktní metody. Definice metody nemusí **obsahovat klíčové slovo abstract**, bude-li v rozhraní jen jedna nedefaultní metoda. Pokud obsahuje další metody, musí nabízet jejich implicitní implementaci. Po implementující třídě tak požaduje implementaci jediné metody. Funkční rozhraní byla **zavedena hlavně kvůli lambda výrazům**. Každý lambda výraz může vystupovat v roli libovolného funkčního rozhraní, jehož abstraktní metoda typově odpovídá danému lambda výrazu. Pokud tedy má metoda funkčního rozhraní například **dva celočíselné parametry a vrací řetězec**, lambda výraz musí také přijímat dva celočíselné parametry a vracet řetězec.

Důvodem, proč ve funkčním rozhraní je možno mít **pouze jednu abstraktní metodu**, je, že kdyby byly v rozhraní třeba dvě abstraktní metody se stejnou návratovou hodnotou i stejným názvem, ale s různými druhy parametrů, tak volající metoda **nebude vědět**, kterou metodu z těch dvou abstraktních metod volat.

Implicitní metody

- Implicitní metody se mohou zdát zbytečností. Dříve, bez výchozích metod, nebylo možné přidat do rozhraní nějakou metodu, aniž by se musely implementovat v dané třídě.
- Řeší častou nutnost použít abstraktní třídu pro definici metod, které by se v implementujících třídách opakovaly.
- Z rozhraní se vyvolají klasicky vytvořením nového objektu a následným zavolání metody

Lambda výrazy

- jsou prvky, kterými se funkcionální programování nejvíce vyznačuje
- Chovají se jako instance funkčního rozhraní, jehož metoda má odpovídající parametry a vrací hodnotu odpovídajícího typu
- Celá tato část kódu se uloží do proměnné funkčního typu a předává se metodám jako hodnota parametru tohoto typu

Použití lambda výrazů a jejich syntaxe je vcelku jednoduchá. Celý tento zápis se skládá z parametrů, pokud jsou nějaké k dispozici, lambda výrazu, ten je značen prostou šipkou (->), a těla výrazu, kde je definováno, co se vlastně bude vracet. Tělem může být jednoduchý výraz, nebo celá posloupnost příkazů. Parametry se vždy udávají vlevo a tělo výrazu vpravo. Syntaxe celého výrazu dost závisí i na počtu parametrů a příkazů v těle výrazu. Do závorek se totiž parametr dávat nemusí, je-li pouze jeden.

V rozhraní je definice metody typu boolean s jedním parametrem. Takže metoda bude vracet buď true nebo false. Dokáže-li překladač odvodit typ parametru, nemusí se uvádět Když je ovšem parametrů více než jeden, musí se dát do kulatých závorek.

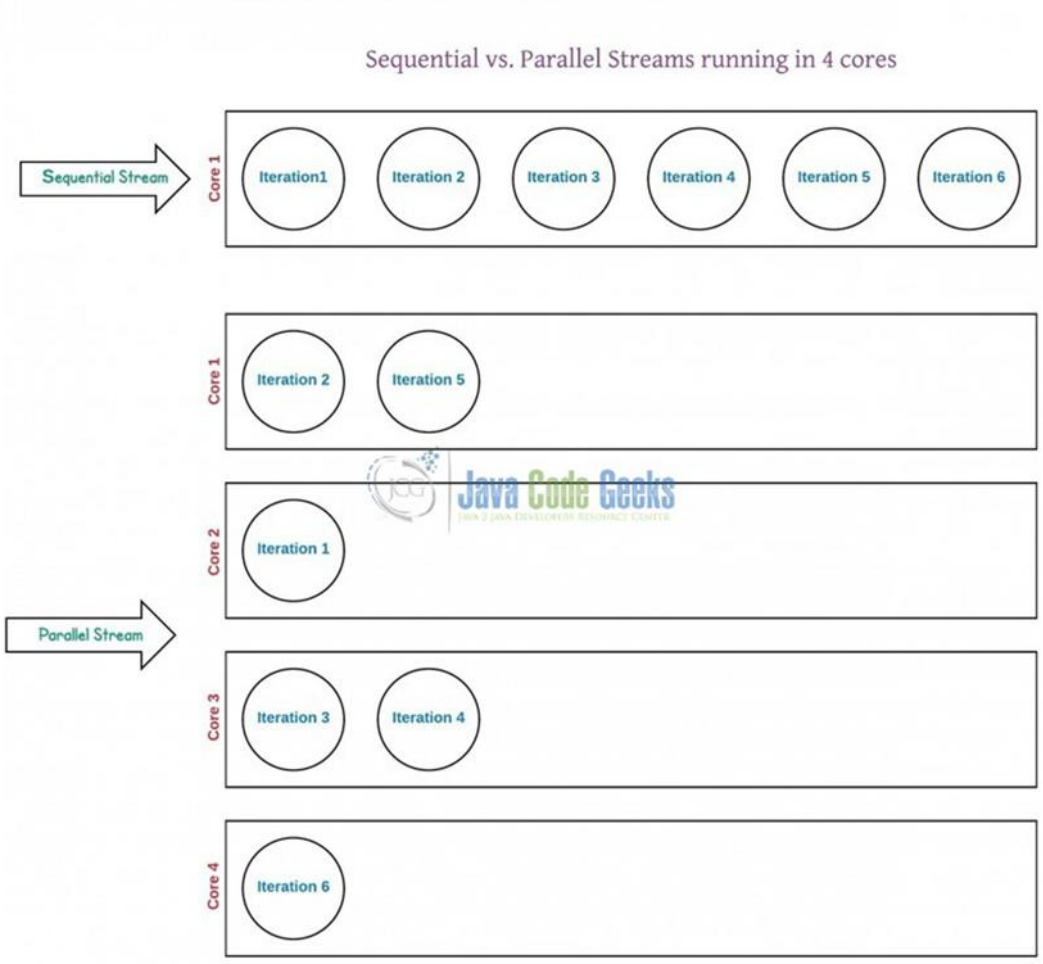
Datovody/ Streamy

Datovody zejména umožňují provádět s daty různé operace. Tyto operace jsou velmi podobné příkazům, které se aplikují na data v jazyce SQL.

Datovody bývají označovány jako ekvivalent kolekcí. Od nich se však liší v několika zásadních věcech.

- Datovod přijatá data pouze zpracuje a pošle dál. To znamená, že si je neukládá a tím nezabírá žádnou paměť.
- Datovod nemění vstupní data. Tudíž mohou být zpracovávána několika procesy zároveň.
- Datovod pracuje na základě odložené vyhodnocovací strategie, čímž šetří výpočetní výkon až do poslední chvíle. Začne se provádět, až je ho třeba
- Datovody se nezajímají o to, zda je vstup dat konečný, nebo nekonečný. Prostě převezmou zdrojová data a dopraví je k požadované operaci

Datovod může data zpracovávat **dvěma způsoby. Sériově a paralelně**. Data v sériovém datovodu jsou zpracovávány **postupně**, to znamená **popořadě jedním vláknem**. Zatímco při paralelním zpracování jsou data zpracovány **paralelně pomocí více vláken neboli podprocesů**. Paralelní zpracování se používá tehdy, nezáleží-li na pořadí, ve kterém budou operace prováděny.



Operace, které datovody obsahují, se dělí na dva druhy, a sice na **průběžné a koncové**. Průběžné operace jsou takové operace, jež se po provedení jako **výstupní hodnota vrací opět** v podobě datovodu. Díky tomu může dojít k takzvanému **zřetězení operací**, kdy na výsledný datovod aplikuje některá z dalších operací. Mezi takové průběžné operace se určitě **řadí filter, sorted, limit, map, skip atd**.

Koncové operace se aplikují pouze tehdy, pokud se **již ukončuje činnost datovodu**. Výstupní hodnota v tomto případě může být **například kolekce, Integer nebo obyčejné vypsání do konzole**. Mezi koncové operace lze zahrnout třeba **count, forEach nebo collect.**

Při volání průběžných metod si datovod **pouze zapamatuje**, ke **které metodě (operaci) má dopravit data**. Celá sekvence se spustí až v okamžiku **zavolání koncové operace**. Teprve ta spustí celou posloupnost datovodů, které **předávají objekty** od jedné operace k druhé, aby byl na konci obdržen požadovaný výstup

10. UML modelování

Charakteristika UML

UML (Unified Modeling Language) je grafický jazyk pro vizualizaci, specifikaci, návrh a dokumentaci programových systémů. Jeho vývoj započal v roce 1994, současná verze 2.5. Spolu s první verzí jazyka UML vznikla i rigorózní metodika RUP (která UML využívá pro grafické znázornění systému a jeho chování).

UML popisuje 14 typů diagramů rozdělených do 3 skupin:

- Statická struktura aplikace
- Dynamické chování
- Organizace a správa aplikačních modulů

UML nabízí standardní způsob zápisu jak návrhů systému včetně konceptuálních prvků (business procesy a systémové funkce), tak konkrétních prvků (příkazy programovacího jazyka, databázová schémata a znovupoužitelné programové komponenty). UML podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů.

UML definuje diagram tříd, objektový diagram, diagram balíčků, diagram komponent, diagram vnitřní struktury, diagram nasazení, diagram případů užití, diagram aktivit, stavový diagram, sekvenční diagram, diagram komunikace, diagram přehledu interakcí a diagram časování.

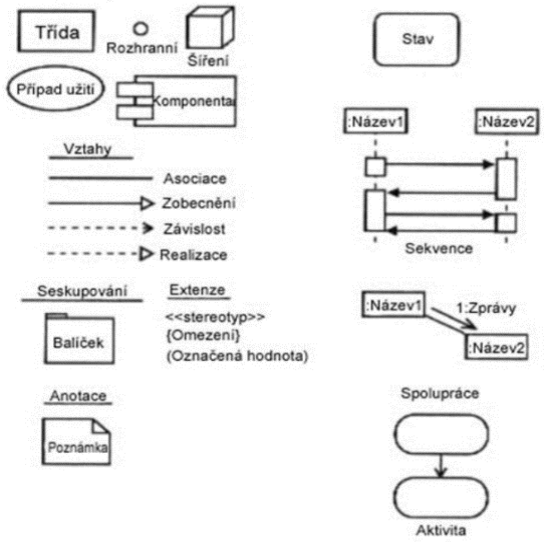


Diagram tříd (Class diagram)

Diagram tříd (Class Diagram) představuje „statický pohled na modelovaný systém“ a jeho úkolem je znázornit typy objektů v systému a jejich vztahy. Návrh tříd, jejich odpovědností a následné vytvoření tohoto diagramu je jedním z prvních a základních kroků analýzy navrhovaného programového systému. Díky tomu, že diagram tříd zachycuje pravidla modelovaného systému, je nejdůležitějším podkladem jak pro forward engineering, tak pro reverse engineering.

Při tvorbě diagramu tříd je nutné vzít v úvahu jeho účel a rozlišit, zda potřebujeme vyjádřit požadavky na modelovaný software nebo získat podrobný popis designu atd. Z tohoto důvodu se rozeznávají tři úrovně modelu tříd – konceptuální, designová a implementační.

Konceptuální (doménový, analytický) model

Vytvářen za účelem analýzy požadavků na software. Obsahuje pouze tzv. byznys třídy (business classes). U jednotlivých tříd se uvádí obvykle jen názvy klíčových atributů a některých klíčových metod. Pokud je diagram vytvářen pouze za účelem znázornění relací mezi třídami, je možné atributy i metody vynechat.

Designový model (model návrhu)

Vychází z modelu konceptuálního, který rozšiřuje a zpřesňuje například viditelnost atributů a metod, datové typy apod. Dále do modelu přidává třídy uživatelského rozhraní (presentation classes) a třídy obsluhující systémové události (control classes). Z jedné třídy v analytickém modelu se tedy může stát v designovém modelu více návrhových tříd.

Implementační model

Zaměřuje se na „grafické zobrazení implementovaného kódu“.

Mezi prvky používané v diagramu tříd lze zařadit:

- třídy (classes)
- asociace (associations)
- rozhraní (interfaces)
- balíčky (packages)

Třída je „abstrakcí objektů se stejnými vlastnostmi, stejným chováním a stejnými vztahy k ostatním objektům.“ Každá třída má popsány vlastnosti a operace (souhrnně označovány jako features), které může provádět, a omezení definující, jak mohou být jednotlivé třídy propojeny. Vlastnosti tříd jsou v UML označovány jako atributy, operace ve

fázi návrhu jako metody (v rámci analýzy se používá označení operace). Třídy jsou vzájemně propojeny pomocí asociací.

Diagram případů užití

Diagram případů užití se používá k popisu chování systému z hlediska uživatele a zachycuje typy uživatelů, kteří se systémem pracují a jaké činnosti v rámci systému vykonávají. Umožňuje znázornit funkční požadavky na systém tím, že popisuje interakci mezi ním a uživateli.

Prvním krokem procesu tvorby softwaru je specifikace softwarových požadavků. Existují dvě skupiny požadavků – **funkční požadavky a nefunkční požadavky**. Případy užití jsou schopny postihnout pouze funkční požadavky, tedy požadavky určující, jaké chování by měl navrhovaný systém nabízet (co by měl systém dělat). Nefunkční požadavky, které představují omezení či vlastnosti systému, zachytit nedokážou. Proto je nutné doplňovat model případů užití i například modelem požadavků, který ale není součástí UML.

V rámci analýzy se vytváří model případů užití, který je tvořen nejen diagramy případů užití, ale měl by obsahovat i specifikaci případů užití a definici aktérů.

Modelování případů užití spočívá v následujících krocích:

- Nalezení hranic systému
- Vyhledání aktérů
- Nalezení případů užití
- Specifikace případů užití
- Určení alternativních scénářů
- Opakování předchozích bodů, dokud nedojde k ustálení případů užití, aktérů a hranic systému

Aktér popisuje uživatele vně systému, který je s ním v interakci. Tímto uživatelem nemusí být pouze fyzická osoba, ale například i jiný systém či hardwarové zařízení. Jeho název pak vyjadřuje jeho roli v systému. Aktéry můžeme rozlišovat na primární a pomocné. Pro vyjádření rolí, které jsou určeny osobám, se obvykle používá figurka, zatímco obdélník zobrazuje roli, kterou hrají jiné systémy. Pokud je figurka použita pro roli systému, je doplněna o stereotyp *<<system>>.*

Případ užití specifikuje část funkcionality systému, kterou využívá aktér a která plní určitý cíl. Je charakterizován množinou scénářů případů užití prováděných za stejným cílem, tedy sekvencí kroků popisující interakci uživatele se systémem.

Vztahy mezi aktéry a případy užití jsou nazývány komunikační asociace či relace (stereotyp *<<communication>>*) a znázorňují mezi nimi plynoucí tok informací.

Mezi případy užití mohou být použity tři typy vztahů:

- **include:** při opakování stejného případu užití na více místech
- **extend:** nadstandardní případ užití při splnění dané podmínky
- **generalizace/specializace:** tento poslední typ vztahu je používán také mezi aktéry, kdy umožňuje znázornit předky či potomky aktéra

Hranice systému (v UML 2.0 označovány jako subjekt) jsou v modelu znázorněny rámečkem kolem případů užití a názvem modelovaného systému.

Sekvenční diagram (Sequence diagram)

Sekvenční diagram je nejvíce používaným diagramem interakcí. Zachycuje grafický průběh zpracování v systému v podobě zasílání zpráv. Sekvenční diagram nejčastěji zobrazuje chování a spolupráci jednotlivých objektů v rámci jednoho případu užití. Pro popis chování jednoho objektu napříč více případy užití se používá stavový diagram.

Zprávy mohou být v sekvenčním diagramu posílány jak mezi jednotlivými objekty, tak i třídami či dokonce aktéry. Proto se prvky, které mezi sebou v diagramu komunikují, nazývají souhrnně **klasifikátory (classifiers).** Z každého

klasifikátoru vede tzv. **čára života (lifeline)**, která reprezentuje, jakým způsoben se instance určitého klasifikátoru účastní interakce.

Diagram komponent

Diagram komponent zobrazuje komponenty, které tvoří aplikaci, systém nebo podnik.

Komponenty (v UML 1.x označovány jako fyzické struktury) reprezentují modulární součásti systému se zapouzdřeným obsahem, které je možné samostatně prodávat i aktualizovat.

Diagram komponent znázorňuje **komponenty (components)** a **závislosti (dependencies)** mezi nimi, popřípadě i rozhraní. Komponenty mohou obsahovat **atributy a metody** a být vnitřně strukturovány.

Diagram aktivit

Diagram aktivit (Activity Diagram) je typem diagramu interakcí, který se používá pro popis procedurální logiky, byznys procesů či pracovních postupů. Umožňuje také graficky modelovat jednotlivé případy užití jako posloupnost akcí.

Diagram aktivit prodělal od svého vzniku mnoho úpravin a i s novou verzí UML 2.0 došlo k jeho změnám. Zatímco v UML 1.x byl chápan jako speciální případ stavového diagramu, UML 2.0 už tyto dva diagramy nijak nespojuje. [Fow2003] Diagram získal novou sémantiku založenou na formálním grafickém modelovacím jazyce Petri Nets (Petriho sítě).

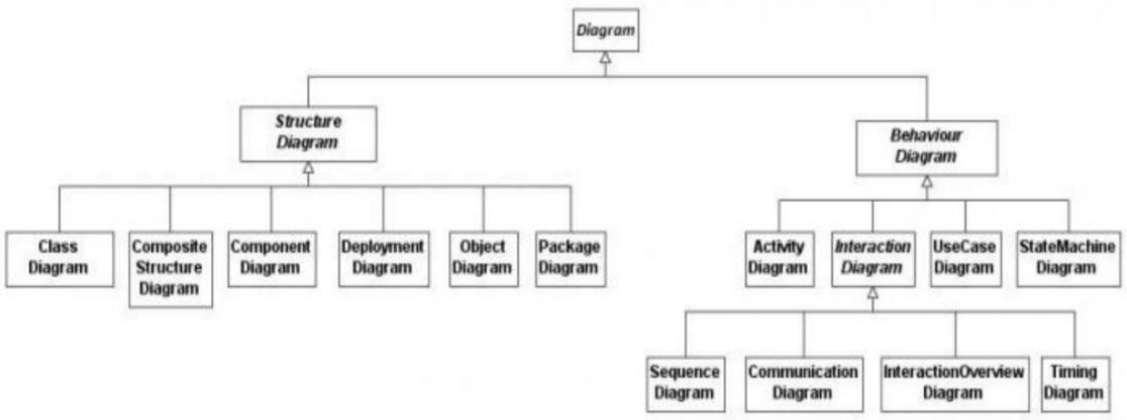
Stavový diagram

Stavový diagram zachycuje jednotlivé stavy objektu a přechody mezi nimi. Stavové diagramy se používají především pro popis chování určitého objektu napříč více případy užití a jejich vznik je spojen už s prvními objektově orientovanými technikami.

Základními prvky stavového diagramu jsou stavy, přechody a události. Pokud to CASE nástroj umožňuje, může být diagram ohraničen rámem s názvem objektu. V případě, že se stavy nepohybují v cyklu, měl by diagram obsahovat **počáteční (initial state) a koncový stav (final state).** Stav (state) je dle situace v životě objektu, během níž objekt splňuje nějakou podmínku, provádí nějakou operaci nebo čeká na událost.

Přechody (transitions) představují podmínky pro přechod objektu z jednoho stavu do druhého. V diagramu jsou značeny linií vedoucí od jednoho stavu k druhému. Jejich popis se skládá ze tří základních částí: událost [podmínka]/ akce. K vykonání uvedené akce a přechodu do dalšího stavu může dojít pouze v případě, pokud je při vzniku události uvedená podmínka (guard) pravdivá. **Událost (trigger signature)** je „specifikací určitého výskytu něčeho v čase a prostoru.“ Pokud není v diagramu uvedena, znamená to, že přechod do dalšího stavu probíhá automaticky.

Rozdělení diagramů dle skupin



Modelování

- popis přesně vymezené části reality, která nás zajímá a která bude obsahem aplikace

- je možné používat slovní popisy i grafické diagramy
- podporuje lepší pochopení požadavků, čisté návrhy a lépe udržovatelný systém

Modelování slouží k:

- prozkoumat možnosti změn bez vlivu na reálné činnosti
- vystopovat příležitosti vylepšení
- odhadnout a měřit účinky zamýšlených změn
- demonstrovat a sledovat průběh analýzy
- předávat nápady jasně a efektivně mezi členy projektového týmu
- model aktuálního stavu x cílového stavu
- analýza vlastností reality na základě modelu
- řízení reality na základě modelu

Vizuální modelování

- způsob přemýšlení o problémech a jejich znázornění pomocí obrázků, efektivní technika

Model

- je konkrétní forma popisu informačního systému
- jedná se o zjednodušení reality
- obvykle nejsme schopni plně pochopit komplexní systém jako celek nebo ani není záměrem modelovat celou realitu
- tvorba modelu pomáhá lépe porozumět systému tím, že ukáže shrnutá pozorování v přehledné formě
- široká škála modelů [umožňují zachycení business procesů, věcné problematiky i architektury systému]
- Model je formulován jako systém, tedy souhrn prvků a jejich vazeb. Konkrétní povaha prvků i vazeb je dána použitým hlediskem (data/operace)
- Zvláštní význam v modelu zaujímají jeho hraniční prvky, tedy prvky, které mají vazby s okolím systému (modelu). Těmito prvky je definována hranice systému, tedy jeho kontext.
- Obsah modelu (souhrn jeho prvků a vazeb) je vždy objektový, tedy každý prvek modelu musí odpovídat některému objektu reálného světa.
- Vnitřní struktura systému (uspořádání prvků) je vždy poplatná struktuře světa.

Model slouží k:

- (grafické) znázornění stávajícího nebo vytvářeného systému,
- specifikace struktury nebo chování systému,
- základ pro (re)konstrukci systému,
- k pochopení problémů,
- komunikace s ostatními zúčastněnými na projektu (uživatel, analytik, programátor atd.)

Kritické faktory úspěchu modelování

- Řešitelé musí být dostatečně seznámeni s problematikou projektu – velmi důležitá je komunikace s klientem
- Musí dodržovat předem domluvenou jednotnou techniku modelování (notace UML, BPMN)
- Dobré rozvržení práce – např. proces jako je fakturace musí být přiřazen specialistovi na finanční procesy (kryje se s klíčovými faktory úspěchu)
- Je nutná kontrola konzistence – při použití více typů modelů – aby každý z nich nevypovídal něco jiného (více typů a pohledů pomáhá k většímu pochopení požadavků na IS/ICT, a proto při nekonzistenci modelů může dojít při budoucím vývoji k problémům)
- Snaha o zachycení všech aspektů, které jsou pro zkoumání systému podstatné a eliminace aspektů nepodstatných

Výhody a nevýhody

Výhody viz předchozí část. Nevýhody:

- Modelování složitých procesů a zachycení všech podstatných věcí a při tom zachování dostatečné čitelnosti modelu
- Modelování je vždy zatíženo nějakým přístupem, který určuje, co má a nemá být ve výsledném modelu a proč.

Objektové modelování

Cíl objektového modelování

- zachytit strukturu reality
- používá se k tomu **jazyk UML** (obecný a univerzální objektově orientovaný modelovací jazyk,
- sloučení několika populárních OO přístupů), složitost je redukována rozdělením do subsystémů a delegováním „odpovědnosti“ na jednotlivé objekty

Komponenty objektového modelování

- **objekt** – prvek, jev, věc, pojem v reálném světě
- **třída** – kategorie, skupina věcí se stejnými vlastnostmi a stejným chováním (nebo podobným), tj. ve smyslu množina objektů stejného typu
- **atribut** – podstatná charakteristika/vlastnost třídy/asociace
- **asociace** – vztah mezi dvěma objekty
- **operace** – metoda, funkce, kterou může objekt vykonávat

Principy

- rozložení reality na jednotlivé prvky (objekty), které spolu interagují
- každý objekt (věc, prvek, jev, pojem v reálném světě) je něčím jedinečný

Diagram tříd

- Prozkoumání problémové domény (typy objektů reality)
- Analýza požadavků IS => ASW (konceptuální => logický model)
- Zachycení detailního návrhu objektově orientovaného SW

Doporučené kroky při objektovém modelování:

- identifikovat objekty a třídy
- připravit slovník dat
- určit asociace a agregace
- určit atributy objektů a linků
- vytvořit hierarchii tříd
- verifikace přístupových cest pro dotazy (ověření úplnosti modelu)
- iterace a upřesňování modelu
- seskupení tříd do modulů

11. Návrhové vzory

- návrhové vzory (design patterns) jsou doporučené postupy řešení často se vyskytujících úloh
- lze je přirovnat k matematickým vzorečkům, nedosazujeme však čísla, ale třídy, rozhraní a objekty
- snižují pravděpodobnost chyb
- vštěpují zásady správného programování
- zestručňují a zkvalitňují komunikaci

- velké SW firmy jejich znalost vyžadují
- existují katalogy návrhových vzorů = nejznámější **je GoF (zkratka Gang of Five** – skupina 5 významných programátorů, popisuje 23 základních, obecně použitelných návrhových vzorů)
- v zápisu návrhových vzorů se používá zakreslení vztahů a postupů pomocí UML diagramů a současně také slovní popis, který dané diagramy doprovází

Definují ověřená řešení určitých problémů návrhu. Obrázky a více info na http://objekty.vse.cz/Objekty/Vzory

Důvod vzniku – potřeba elegantního, jednoduchého a znovupoužitelného řešení.

Je třeba najít vhodné objekty, definovat rozhraní tříd, definovat hierarchii dědičnosti, definovat vztahy mezi třídami.

Cíl:

- sepsat katalog obecných interakcí, které byly vícekrát použity
- chceme získat nezávislost tříd – volné vazby
- u dědičnosti má potomek přístup k metodám a nesoukromým proměnným předka, když je na vrcholu hierarchie třída s definovanou implementací, tak potomci jí mají také.

Formát vzoru (jednoduchý):

- název problému
- problém (kdy se má vzor používat)
- řešení (prvky řešení)
- důsledky (důsledky použití vzoru)
- související vzory

Návrhové vzory podle Pecinovského (které nejsou součástí GoF):

Jednoduchá tovární metoda – metoda, která umí vracet instanci deklarovaného typu (buď jí vytvoří, nebo najde nějakou již existující). Normální konstruktor umí instanci jen vytvořit, a navíc umí vytvářet jen instance své třídy + má další omezení.

- Přepravka (Messenger) – pokud potřebuji, aby metoda vracela více hodnot najednou, tak je můžu dát do přepravky – to je třída, jejíž instance přenáší požadované hodnoty ve svých atributech. Tyto atributy bývají veřejné, aby bylo možné přistupovat k hodnotám přímo. Samozřejmě ale mohou být privátní, a tedy přístupné jen přes příslušné metody. Přepravka tedy může mít i metody. Každá přepravka má předem připravené atributy, které se jen naplňují – pro každý typ přenášených hodnot tedy musí existovat zvláštní přepravka (např. přepravka na rozměry má atributy x a y).
- Knihovní třída (Utility) – třída, která slouží jako úložiště (knihovna) často používaných metod – je to lepší než mít v každé třídě tu samou implementaci těchto metod. Metody v této třídě bývají statické. Obvykle nemá smysl vytvářet instanci této třídy, proto její konstruktor bývá privátní, bezparametrický a s prázdným tělem.
- A další – kontejner, výčtový typ, neměnné objekty, služebník (servant), prázdný objekt (null object) …

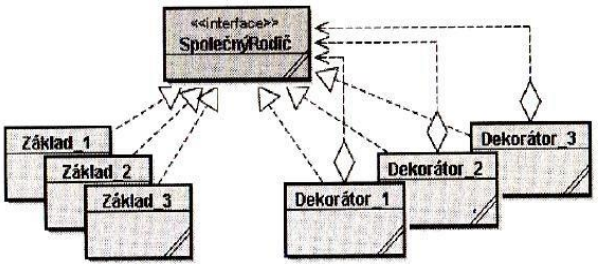
Členění podle GoF:

- **Structural Patterns** (strukturální vzory) – zaměřujících se na možnosti uspořádání jednotlivých tříd nebo komponent v systému
- **Creational Patterns** (vzory pro vytváření objektů) – řeší problémy související s vytvářením objektů v systému
- **Behavioral Patterns** (vzory chování) – zajímají se o chování systému

Strukturální vzory

- **Adapter (adaptér)**
    - Potřebujeme existující rozhraní objektu přetransformovat na rozhraní, které v danou chvíli potřebujeme.

    - Může se implementovat třemi různými způsoby. Vždy jde ale o to, že adaptér tvoří prostředníka mezi klientem volajícím určitou metodu a třídami, které tuto metodu implementují, a přesměrovává volání od klienta na některou z těchto tříd.
    - Příklady:
        - Obalové třídy primitivních datových typů v Javě – abychom mohli používat metody, které se dotazují na jejich hodnoty, musí být primitivní datové typy obalené adaptérem.
        - Zajištění zpětné kompatibility systému – požadavky se musí zpracovávat podle toho, jaké rozhraní klient očekává.
- **Bridge (most)**
    - zavádí rozhraní, které odděluje abstrakci zprostředkovávající nějakou službu od její implementace, takže lze obojí měnit nezávisle na sobě
    - příklad: okno v GUI – abstrakce okna a její funkcionalita x specifická implementace na konkrétním OS
- **Composite (skladba)**
    - slouží pro jednotné vyjádření hierarchií typu celek-část
    - příklad – stromová struktura
- **Decorator (dekorátor)**
    - dynamicky připojuje další funkcionalitu k objektu – pokud chci, aby více objektů získalo určitou
    - funkcionalitu, a nechci jí v každém implementovat zvlášť, tak je zabalím do určité třídy. Obalující třída má na starosti jen požadovanou funkcionalitu a zbytek deleguje na obalený objekt.
    - flexibilní alternativa dědění
    - někdy potřebujeme přidat odpovědnosti jen jednotlivým objektům, nikoli celé třídě
    - často se používá v Javě v GUI – např. při návrhu GUI potřebujete přidat ohraničení nebo rolování k určité komponentě. Řešení je, že vložíte komponentu do dekorátoru.
    - Implementace – mám několik tříd, kterým potřebuji dodat určitou funkcionalitu. Tak jim vytvořím společné rozhraní, které implementují nejen tyto třídy, ale i dekorátory. Stejně tak to můžu udělat s pomocí abstraktní třídy, která bude předkem těchto tříd, a tak může implementovat nějakou funkcionalitu, která je pro všechny dceřiné třídy společná.



- **Facade (fasáda)**
    - cílem je zjednodušit (sjednotit) rozhraní
    - snižuje počet tříd, se kterými musí uživatel komunikovat
- **Flyweight (muší váha)**
    - vzor vhodný pro situace, kdy vzniká mnoho malých objektů, u kterých je možné podstatnou část jejich stavu sdílet nebo ho nahradit výpočtem
- **Proxy (zástupce)**
    - zavádí zástupce, který odstiňuje objekt od jeho uživatelů a sám řídí přístup uživatelů k danému objektu

Vzory pro vytváření objektů

Řeší problémy související s vytvářením objektů v systému:

- snaží se oddělit vytváření objektů do samostatné třídy/tříd
- konkrétní třída, od níž se vytváří instance, se určuje až za běhu programu
- třeba zajistit správný počet objektů

Tovární metoda (Factory Method)

- několik tříd, které mají obvykle společného předka a poskytují různé služby
- dovoluje za běhu programu vytvořit instanci některé z těchto tříd
- definuje rozhraní pro vytváření objektu; přenechává rozhodnutí, od které třídy vytvořit objekt, na své podtřídy

Abstraktní továrna (Abstract Factory)

- při vytváření více objektů, které spolu souvisí
- poskytuje rozhraní pro vytváření rodiny příbuzných objektů bez nutnosti specifikovat konkrétní třídy
- složená třída obsahující stejné složky vytvářené různě v závislosti na aplikaci
- abstraktní továrna je založena na myšlence, že se jako parametr do konstruktoru předá třída (továrna) se standardním rozhraním – to je pak použito konstruktorem, ale jeho metody jsou implementovány odlišně pro každý typ továrny

Stavitel (Builder) – podobný abstract Factory. Máme různé objekty s podobným procesem konstrukce.

Prototype – vytvoření kopie existující objektu namísto vytváření nové třídy (v Javě rozhraní Cloneable).

Jedináček (Singleton)

- cílem je zabránit vícenásobnému spouštění konstruktoru – chceme, aby třída měla pouze jedinou instanci
- řešení:
    - použít statickou proměnnou třídy, která bude obsahovat informaci o tom, zda již byla vytvořena instance třídy
    - definovat privátní konstruktor třídy
    - definovat statickou metodu getInstance(), která bude vracet instanci třídy
- příklady – jediné připojení k databázi, správce tiskových úloh

Vzory chování

Mediator (=prostředník)

- Definujeme objekt prostředníka, který zprostředkovává veškerou komunikaci objektů
- Vzájemné závislosti objektů se tak omezí na závislost na prostředníku
- u telefonů – obdobné řešení – telefony také nejsou spojeny každý s každým, ale spojují se prostřednictvím ústředny
- většinou implementován pomocí vzoru Observer

Observer (=pozorovatel)

- Problém: na stavu jednoho objektu závisí jiné objekty. Při změně stavu jednoho objektu je třeba informovat objekty na něm závislé. Tyto objekty jsou volně vázané – objekt měnící svůj stav informuje ostatní, ale je nezávislý na jejich vnitřním uspořádání.
- řešení – objekt měnící stav = **subjekt** – informuje objekty u něj zaregistrované jako pozorovatele/posluchače = **observer**
    - pozorovaný objekt (subjekt) udělám potomkem třídy Observable a zařídím, aby ve správnou chvíli informoval pozorovatele.
    - pozorovatele nechám implementovat rozhraní Observer a definuji v nich metodu update (aktualizuj), kterou subjekt při dané změně stavu zavolá.
- příklad:
    - změní se zaškrtnutý RadioButton
    - v adventuře byly na změně místnosti závislé věci v dané místnost

Přínos návrhových vzorů

- popisují opakující se problém spojený s návrhem softwaru a poskytují jeho řešení,
- zachycují a dokumentují již existující a praxí ověřený návrh,
- umožňují pracovat na vyšší úrovni abstrakce,
- definují jazyk pro popis návrhu,
- dokumentují architekturu,
- stavební prvky, z nichž se dají stavět komplexní návrhy,
- zajišťují vyšší kvalitu SW

Výhody používání vzorů

- zvýšení znovupoužitelnosti a produktivity
- řízení znalostí
- rychlejší a efektivnější vývoj – vývojáři se zaměřují na to co, vytvářet a ne, jak to vytvářet
- automatizované generování kódu pro vzory
- zvýšení kvality kódu

12. Metodiky budování IS

Metodika představuje v obecném smyslu souhrn metod a postupů pro realizaci určitého úkolu
KDO?KDY?CO?JAK?PROČ?

- Metodiky zabývající se údržbou a vývojem IS bývají označovány jako metodiky vývoje IS/ICT.
- Kromě vývoje je rovněž důležité také nasazení hotového řešení, rozšíření řešení a integrace stávajících řešení, jsou tyto metodiky označovány jako metodiky budování IS/ICT.

Metodiky budování IS/ICT = metodiky vývoje IS/ICT + metodiky provozu IS/ICT

Vývoj a provoz infformačního systému je obtížné oddělit, neboť některé části IS (subsystémy, moduly, komponenty, služby aj.) jsou v daném okamžiku v provozu, některé jsou rozvíjeny, některé jsou vytvářeny nově a všechny musí být dohromady integrovány. Proto metodiky budování IS/ICT pokrývají jak oblast vývoje, tak oblast provozu IS/ICT.

Problematikou provozu IS/ICT se však nezabývají v celé šíři, neboť je tato oblast řešena v rámci metodik pro řízení informatiky (například COBIT, ITIL).

Termín metodika budování IS/ICT je vymezen následovně: Metodika budování IS/ICT definuje principy, procesy, praktiky, role, techniky, nástroje a produkty používané při vývoji, údržbě a provozu informačního systému, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení. Určuje kdo, kdy, co, jak a proč dělat během vývoje a provozu IS.

Metodika napomáhá

- k tomu, aby IS byl přínosem pro uživatele (organizaci, zákazníka, …)
- k tomu, aby byly provedeny všechny potřebné činnosti tvorby IS, a to ve správné časové posloupnosti
- k tomu, aby IS byl srozumitelně dokumentován
- k dobré organizaci práce na projektu
- k optimalizaci spotřeby zdrojů při tvorbě a provozu IS.

Základní požadavky efektivní využitelnosti metodik:

- musí jasně deklarovat soubor hodnot, na kterých je založena,
- musí určovat postup řešení, aby bylo možné celý proces vývoje IS/ICT plánovat,
- musí se zabývat všemi faktory (dimenzemi), které tvorbu a provoz IS ovlivňují
- musí určovat priority řešení (co a kdy je důležité),
- měla by doporučovat metody, techniky a nástroje, které je vhodné využít v jednotlivých fázích řešení.

Kritérium	Kategorie/ vysvětlení
Zaměření metodiky	• Globální – zaměřují se na budování celopodnikového IS/ICT • Projektové – pouze dílčí oblast (výroba, finance…) – sem patří většina metodik
Rozsah metodiky	• Fáze životního cyklu IS/ICT – globální strategie, informační strategie, úvodní studie, globální analýza a návrh, detailní analýza a návrh, implementace, zavádění, provoz a údržba • Role – sponzor, vedoucí projektu, koncový uživatel, analytik, návrhář, tester, dokumentátor, expert na doménu • Dimenze – hardware, technologie, data-informace, funkce-procesy, uživatelské rozhraní, pracovní dimenze, organizační/legislativní, ekonomická

Váha metodiky	• podrobnost – do jaké hloubky se daným tématem zabývá • přesnost – jak je dané téma zpracováno • relevance – zda se zabývá určitým tématem • tolerance – tolerance odchylek • měřítko – míra zaostření (několik položek může být shrnuto do jednoho celku)
Typ řešení	• vývoj nového řešení • rozvoj a rozšíření stávajícího řešení • integrace řešení • customizace a implementace typového řešení • užití řešení například formou ASP
Doména	• Předmětná oblast – ERP, CRM, SCM, BI, EAI...
Přístup k řešení	• Strukturovaný vývoj • Objektově orientovaný vývoj • Rychlý vývoj aplikací – Rapid Application Development (RAD)

Rigorózní metodiky

V současnosti můžeme sledovat dva hlavní proudy v metodických přístupech, které jsou označovány jako rigorózní metodiky a agilní metodiky (popsány v otázce č. 9). Hlavním kritériem, které tyto dva proudy odlišuje, je „Váha metodiky“, ale liší se i dalšími hledisky.

V této kapitole jsou charakterizovány rigorózní metodiky, které vycházejí z přesvědčení, že procesy při budování IS/ICT lze popsat, plánovat, řídit a měřit. Snaží se podrobně a přesně definovat procesy, činnosti a vytvářené produkty, a proto bývají často velmi objemné. Rigorózní metodiky jsou zpravidla založeny na sériovém (vodopádovém) vývoji. Existují ale také rigorózní metodiky založené na iterativním a inkrementálním vývoji.

Příkladem těchto metodik jsou OPEN, Rational Unified Process (RUP), Enterprise Unified Process (EUP). V rámci rigorózních metodik tvoří samostatnou kategorii metodiky pro hodnocení softwarových procesů (Software Process Assesment). Metodiky hodnocení softwarových procesů jsou založeny na přesvědčení, že kvalita procesu určuje kvalitu produktu, a proto popisují postupy, které umožňují hodnotit úroveň zralosti procesů při vývoji software. Nejznámější z těchto metodik je Model zralosti (Capability Maturity Model for Software).

Charakteristika rigorózních metodik

- těžké metodiky – podrobné, hodně formalit, direktivní řízení
- předpokládají opakovatelnost procesů, možnost definovat všechny požadavky na řešení předem
- příklady – OPEN, RUP, EUP, OOSP
- metodiky pro hodnocení SW procesů (Software Process Assesment) – Capability Maturity Model (CMM)

Metodika OPEN = Object–oriented Process, Environment and Notation

OPEN je veřejně přístupná metodika podporující celý životní cyklus vývoje IS/ICT. Je zaměřena zejména na vývoj objektově orientovaných a komponentových aplikací. Metodika OPEN definuje procesní rámec, známý pod názvem OPEN Process Framework (OPF). Jde o procesní metamodel, ze kterého mohou být generovány instance specifické pro organizaci.

OPEN je flexibilní, může se přizpůsobit jak doméně, tak konkrétnímu projektu, a zohlednit dovednosti členů týmu, kulturu organizace, požadavky specifické pro každou doménu. OPEN může být použit pro malé projekty, stejně jako pro velké, klíčové projekty.

Metodika RUP = Rational Unified Process

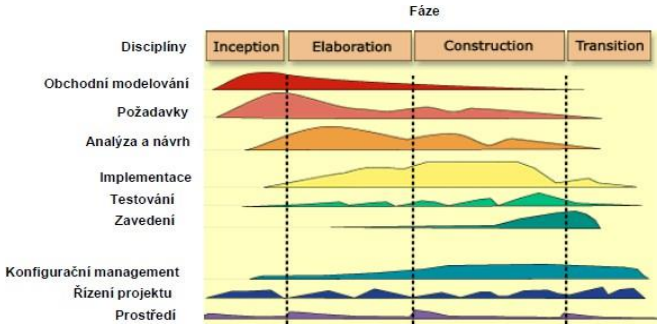
Metodika Rational Unified Process je založena na tzv. nejlepších praktikách softwarového vývoje:

- iterativní vývoj
- řízení požadavků
- použití komponentové architektury
- vizuální modelování (UML, CASE nástroje…)

- kontrola kvality software
- řízení změn

Životní cyklus software je rozdělen na cykly. Předmětem každého cyklu je nová verze produktu. Jeden vývojový cyklus je v RUP rozdělen do 4 po sobě jdoucích fází:

- Počáteční fáze (Inception)
- Elaborační fáze (Elaboration)
- Konstrukční fáze (Construction)
- fáze Nasazení (Transition)



1. **Počáteční fáze** = Definice cílů projektu, požadavků, sestavení harmonogramu (plán iterací), odhad nákladů projektu a rizik.
2. **Elaborační fáze** = Má za cíl definovat architekturu systému. V této fázi by měl být vytvořen prototyp.
3. **Konstrukční fáze** = Návrh a realizace systému včetně testování.
4. **Fáze nasazení** = Zajišťuje, aby uživatelé mohli systém používat. Součástí této fáze je školení uživatelů, předání dokumentace, vytvoření help-desk atd. Každá fáze je uzavřena milníkem – časovým okamžikem, ve kterém musí být splněny cíle fáze a dochází k rozhodování. Podstatným nedostatkem metodiky RUP je: zaměřuje pouze na vývoj řešení, ne na jeho provoz a údržbu.

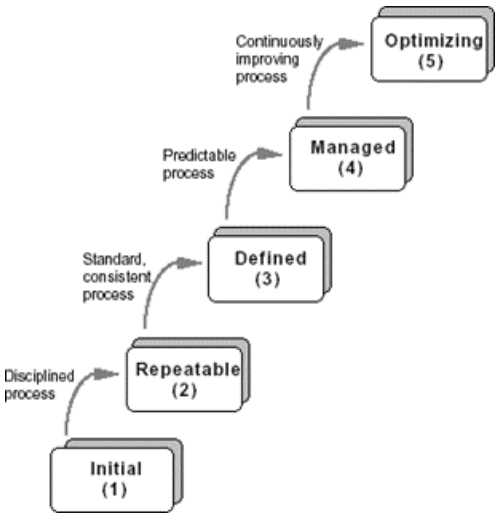
projektu definice

Model zralosti SW – Capability Maturity Model for Software (SW–CMM)

Nejznámějším příkladem hodnocení softwarových procesů je Model zralosti SW (Capability Maturity Model for Software). Zralost softwarového procesu (Software Process Maturity) je dosažena, pokud je určitý proces explicitně definován, řízen, měřen, kontrolován a je efektivní. Zralost softwarových procesů vede ke zvýšení produktivity a kvality a zvyšuje se výkon procesu.

- označovaný zkratkou SW-CMM.
- vyvinut v Institutu pro softwarové inženýrství (Software Engineering Institute – SEI) za
- účelem hodnocení dodavatelů softwarových řešení pro ministerstvo obrany USA – v r. 1995
- Integrační model zralosti (Capability Maturity Model Integration, CMMI) od r. 2000

1) **Počáteční úroveň (initial)**
   - softwarové procesy jsou náhodné a chaotické
   - organizace nemají stabilní prostředí pro vývoj a údržbu software, reagují pouze na vzniklé problémy.
2) **Opakovatelná úroveň (repeatable)**
   - organizace mají definovány a zavedeny postupy řízení projektu.
   - softwarový proces je disciplinovaný
3) **Definovaná úroveň (defined)**
   - organizace má definovány, dokumentovány a standardizovány procesy pro řízení i softwarově inženýrské činnosti, které jsou navzájem integrovány v rámci celé organizace
   - softwarový proces je standardní a konzistentní
4) **Řízená úroveň (managed)**
   - organizace má stanoveny detailní metriky softwarových procesů i kvality produktu
   - softwarový proces je predikovatelný
5) **Optimalizovaná úroveň (optimizing)**



- organizace má vytvořeny podmínky pro kontinuální zlepšování procesů.

Agilní metodiky

Agilní metodiky začaly vznikat v druhé polovině 90. let jako reakce na nedostatky tradičních rigorózních metodik. Umožňují vytvořit řešení velmi rychle a pružně jej přizpůsobovat měnícím se požadavkům. Tyto „lehké“ metodiky vývoje SW vychází z myšlenky, že jedinou cestou, jak otestovat software, je co nejrychleji ho vyvinout (nebo jeho část), předložit zákazníkovi a na základě zpětné vazby jej upravit. Jednotlivé agilní metodiky používají rozdílné techniky, ale jsou založeny na společných principech a hodnotách. V únoru 2001 byl podepsán „Manifest agilního vývoje softwaru“ a byla vytvořena „Aliance pro agilní vývoj softwaru“.

Největší použitelnost se předpokládá u:

- výzkumných projektů
- time-to-market
- menších týmů

Principy agilních metodik

Výše zmíněný „Manifest agilního vývoje softwaru deklaruje čtyři hodnoty, přičemž tučně vyznačené prvky mají větší relativní význam než prvky vedle nich. „Odhalili jsme lepší způsob vývoje softwaru, sami jej používáme a chceme pomoci i ostatním, aby jej používali.

Z tohoto pohledu **dáváme přednost**:

- individualitám a komunikaci před procesy a nástroji
- provozuschopnému softwaru před obsažnou dokumentací
- reakci na změnu před plněním plánu
- spolupráci se zákazníkem před sjednáváním kontraktu.

Na základě tohoto manifestu bylo definováno **10 hlavních principů** agilních metodik:

- včasná a kontinuální dodávka softwaru s hodnotou pro zákazníka
- změna požadavků i v průběhu vývoje
- každodenní spolupráce uživatelů a vývojářů
- podpora motivovaných jedinců
- osobní komunikace
- fungující software
- „zdravý vývoj“
- perfektní technické řešení i návrh
- jednoduchost řešení, maximalizace množství neudělané práce
- samoorganizující se týmy

Charakteristika agilních metodik společné principy:

- iterativní vývoj s velmi krátkými iteracemi,
- zaměření na fungující SW, který má hodnotu pro zákazníka,
- lidé jsou prvořadým faktorem – důraz na spolupráci a komunikaci,
- tolerantní ke změnám,
- automatizované testování.

Místo procesů praktiky:

- **Proces** – je popisován v manuálech, pohlíží na lidi jako na sekundární, zaměřuje se na explicitní (popsanou) znalost
- **Praktika** – vychází ze zkušenosti, soustředí se primárně na lidi, soustředí se na interní „tacit“ znalosti

přesun zodpovědnosti za požadavky na zákazníka:

- zákazník určuje a mění priority funkcí

spolupráce zákazníků a vývojářů (rigorózní metodiky x agilní metodiky):

- **Rigorózní metodiky**
  - Nevěřím ti, že uděláš práci správně, tak tě musím neustále sledovat a kontrolovat.
  - standardizují lidi v organizaci
  - snaží se vykázat lidi do role zaměnitelné součástky
- **Agilní metodiky**
  - Věřím ti, že uděláš práci dobře, a tak budeme spolupracovat, abychom dosáhli výsledku.
  - využívají individualit a silných stránek lidí

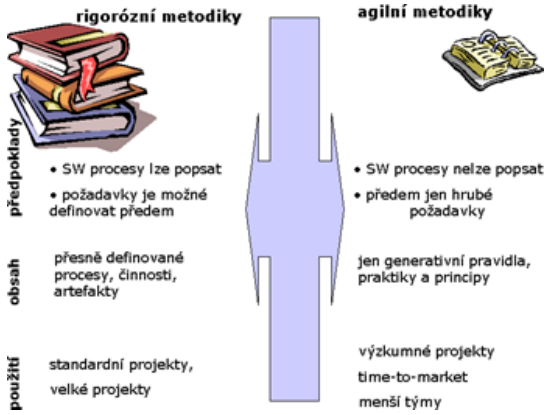
Rozdíl mezi metodikami

Rigorózní metodiky:

- požadavky, procesy je třeba specifikovat předem a změnám se snažíme zabránit
- hodně formalit, dokumentace, direktivní řízení
- založeno na nedůvěře – na všechno musí být papíry, všechno podloženo smlouvami atd.
- příklady: OPEN, RUP, EUP

Agilní metodiky:

- individualita, pozitivní přístup ke změnám
- důležitější, než dokumentace je provozuschopný software
- klade důraz na přidanou hodnotu pro zákazníka
- spolupráce se zákazníkem – častá komunikace



Porovnání rigorózních a agilních metodik		
hledisko	**Rigorózní metodiky**	**Agilní metodiky**
náplň metodiky	procesy, zaměřují se na explicitní znalost a pohlíží na lidi jako na sekundární faktor	praktiky, zaměřují se na „tacit“ znalosti, chápou lidi jako klíčové faktory úspěchu
Podrobnost metodiky	procesy a činnosti jsou popsány velmi podrobně	definována tzv. sotva dostatečná metodika, která se zaměřuje na činnosti, které vytvářejí hodnotu, a eliminuje činnosti, které hodnotu nepřinášejí
kvalita	zaměření na kvalitu procesů a předpoklad, že kvalitní procesy povedou ke kvalitnímu výsledku	zaměření na hodnotu pro zákazníka a vysokou kvalitu produktu
předvídatelnost	předpokládá předvídatelnost budoucnosti, důraz na anticipaci (sběr požadavků předem, plánování předem)	předpokládá nepředvídatelnost budoucnosti, důraz na adaptaci na změny (přírůstkové shromažďování požadavků, plánování pro iteraci)
změny	změny podléhají řízení změn a je snaha změny minimalizovat	snaha změny umožnit a využít je, umožňují zákazníkům přehodnotit své požadavky s ohledem na nové znalosti
Definovatelnost procesu vývoje software	vývoj software je definovaný proces, je možné jej bez problémů opakovat	vývoj software je empirický proces, nemůže být konzistentně opakován, ale vyžaduje konstantní monitorování a adaptaci
hodnota pro zákazníka	předpoklad, že dobré procesy vedou k dobrým výsledkům, je příliš zaměřen na vlastní procesy, ne na výsledky pro zákazníka.	nejvyšší prioritou je uspokojovat zákazníka
Participace zákazníka na	jen v počátečních a koncových fázích, po podpisu dokumentu specifikace	přesun nositele řízení z týmu na zákazníka, zákazník je řídícím subjektem během celého

projektu	požadavků řízení přebírá tým technologických pracovníků	projektu, při každé iteraci zákazník může měnit priority funkcí
rozsah řešení	vývojáři se snaží do systému zabudovat všechny funkce, které by mohl zákazník v budoucnu potřebovat.	pouze požadované funkce, požadavek minimalizace
vztah zákazník - vývojář	zajištěn smluvně, nedůvěra	důvěra a spolupráce
lidský faktor	sekundární, dokumentačně zaměřené procesy se snaží vykázat lidi do role zaměnitelné součástky	primární, využívá individualit a silných stránek lidí
kvalifikace lidí	stačí standardní jedinci	důraz na schopnosti, znalosti a dovednosti lidí
specialisté x generalisté	požadavek úzké specializace lidí	požadavek integrace znalostí a stálé kooperace, sdílení znalostí v týmu, týmové řešení problému, spíše generalisté než specialisté
způsob řízení	tradiční způsob řízení je formován na základě nedůvěry, direktivní řízení, kontroly	vůdcovství a spolupráce, je formováno na důvěře a respektu
Význam programování při vývoji SW	důraz a hodnota jsou kladeny na architekturu, požadavky a návrh, kódování a testování jsou chápány jako činnosti s nízkou „konstrukční“ hodnotou.	důraz na programování jako činnost přinášející hodnotu
jednoduchost	spíše složité řešení, které se snaží obsáhnout i budoucí požadavky	důraz na jednoduché řešení žádné zabudovávání budoucích požadavků
jednoduchá x složitá pravidla	metodiky se snaží popsat vše, s čím se může vývojový tým setkat.	obsahují generativní pravidla – minimální množinu věcí, které musíte dělat ve všech situacích.
modelování	velký důraz na modelování, zejména modelování předem – big design in front of , potom se zmrazí požadavky	agilní modelování, při modelování nejde o model jako takový, ale o akt modelování, smyslem modelování je komunikace
forma komunikace	převážně písemná	důraz na komunikaci tváří v tvář
dokumentace	rozsáhlá dokumentace	podstatná není dokumentace, ale pochopení
způsob vývoje	spíše vodopádový, případně iterativní a přírůstkový s dlouhými iteracemi	přírůstkový vývoj s velmi krátkými iteracemi
ekonomika	zdroje bývají proměnnou veličinou, která zpravidla roste	snaha vždy realizovat nejvyšší hodnotu z daných peněz, cílem je hodnota pro zákazníka, ne perfektní systém, hodnota je kombinací funkcí produktu, které odpovídají potřebám zákazníka v určitý čas a za určitou cenu

Příklady metodik

Dynamic Systems Development Method (DSDM)

DSDM kombinuje přístup rychlého vývoje aplikací (RAD) s objektově orientovaným vývojem. Klade velký důraz na kvalitu řešení. Metodika je velmi dobře dokumentována a řízeným způsobem rozšiřována. Je to projektová metodika, zaměřená zejména na softwarově-inženýrskou oblast, méně se zabývá oblastí řízení. Je zaměřena pouze na vývoj nového řešení. Základní technikou používanou při analýze a návrhu je prototypování. Přínosem metodiky je řízení jejího rozvoje, propagace, školení a implementace.

Metodika DSDM je postavena na 9 principech:

- aktivní zapojení uživatele
- tým s rozhodovací pravomocí
- časté dodávky produktů
- klíčovým kritériem pro přijetí dodávky je podpora podnikových cílů

- iterativní a inkrementální vývoj jako nástroj postupného přibližování k žádoucímu řešení
- změny v průběhu vývoje
- definice požadavků na hrubé úrovni
- testování v průběhu celého životního cyklu

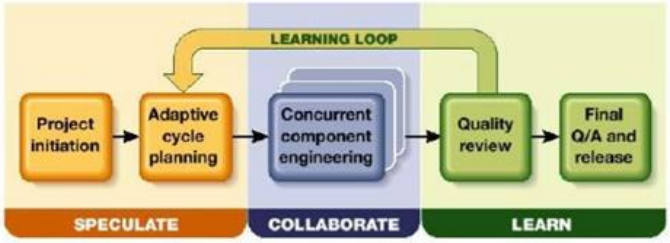
Metodika DSDM rozděluje proces vývoje do třech hlavních fází – Funkční model (Functional Model), Návrh (Design and Build) a Implementace (Implementation), kterým předchází Studie proveditelnosti (Feasibility Study) a Byznys studie (Business Study). Hlavní fáze probíhají iterativně. Obsahem fáze Funkční model je sběr a prototypování funkčních požadavků. Většina požadavků je dokumentována jako prototypy. Ve fázi Návrh jsou prototypy zpodrobňovány, tak aby podporovaly všechny požadavky (i nefunkční) a je navrhováno řešení. Náplní fáze Implementace je realizace navrženého řešení, zhodnocení projektu, školení uživatelů a další činnosti.

Adaptive Software Development (ASD)

ASD je „lehkou“ metodikou, která se zabývá jak oblastí softwarově inženýrskou, tak oblastí řízení. Metodika ASD představuje tzv. „filozofické zázemí“ pro agilní metodiky. Nahrazuje statický životní cyklus „Plánování-Návrh-Realizace“ (Plan-Design-Build) dynamickým „Spekulace-Spolupráce-Učení“ (Speculate-Collaborate-Learn), kde naučení klade největší důraz. Je vhodná pro projekty vyznačující se vysokou rychlostí, změnami, neurčitostí

Základem dynamického cyklu je kontinuální učení

- Hybnou silou jsou neustále změny
- V klasickém cyklu je problematická oblast plánování (odchylky jsou chyby) – spekulace proto dává více prostoru pro změny



Lean Development (LD)

Metodika je podobně jako Scrum zaměřena zejména na řízení vývoje softwaru a na řízení rizik. Méně se pak zabývá softwarově inženýrskou oblastí. Zaměřuje se spíše na strategickou úroveň s vazbou na podnikouvou strategii. LN je nástrojem přechodu na podnikání tolerantní ke změnám (change tolerant business) a rizikové podnikání (risk entrepreneurship).

Znaky:

- celulární programování – programátoři jsou rozděleni do týmů, které řeší jednotlivé problémy odděleně
- tlačení rozvrhem – důraz na rozvržení prací a dodržení termínů
- total quality management – souvislý proces řízení kvality, filosofie stálého zlepšování všech činností
- rapid setup – co nejrychlejší příprava k samotné práci
- týmový vývoj

Feature Driven Development (FDD)

Vývoj začíná vytvořením celkového modelu a pokračuje posloupností dvoutýdenních iterací, ve kterých se provádí návrh i realizace pro jednotlivé užitné vlastnosti. Užitná vlastnost (feature) je malý výsledek užitečný pohledu zákazníka. FDD je projektová metodika zaměřená na objektově orientovaný vývoj nového softwaru. Do určité míry podporuje procesy, modelování a používání CASE nástrojů.

Výše zmíněná užitná vlastnost neboli feature se skládá z 5 procesů:

- vytvoření celkového objektového modelu (Develop an Overall Model)
- sestavení seznamu užitných vlastností (Build a Features List)
- plánování pro užitnou vlastnost (Plan by Feature)
- návrh pro užitnou vlastnost (Design by Feature)
- realizace pro užitnou vlastnost (Build by Feature)

Crystal metodiky

Crystal je soubor metodik pro různé druhy projektů, které se liší důležitostí a velikostí týmu a tím, na co je projekt optimalizován. Všechny metodiky mají společné hodnoty a principy, liší se použitými technikami, rolemi, nástroji a standardy. Jsou zaměřeny na objektově orientovaný vývoj nového řešení.

Scrum

Metodika Scrum je jazykem vzorů (pattern language) a je zaměřena hlavně na oblast řízení projektu. Název je odvozen ze skrumáže neboli mlýna v ragby, aby byla zdůrazněna adaptibilita, rychlost a schopnost samoorganizace této metodiky. Vývoj probíhá ve Sprintech (30denních iteracích), ve kterých je dodávána vybraná množina užitných vlastností. Sprintů bývá 3-8. Klíčovou praktikou je používání tzv. Scrum Meetings.

Jak probíhá Sprint:

- všechna práce se dělá uvnitř sprintu, Sprint je 30denní iterace
- na začátku sprintu se koná Sprint Planning Meeting: trvá max. 8 hodin, cíl – definovat Sprint Backlog
    - každý den se koná Scrum Meeting – 15–30 min, probírá se:
        - co se dokončilo
        - nové úkoly
        - omezení a překážky pro plnění úkolů
- na konci sprintu se koná Sprint review meeting , trvá 4 hodiny – zhodnocení
- Role:
    - Scrum master: zodpovídá za Scrum proces, jeho správnou implementaci a maximalizaci užitku, vede scrum meetingy, je členem týmu.
    - Product owner: spravuje seznam požadavků tak, aby maximalizoval hodnotu projektu)
    - Tým: skupina lidí, kteří se sami řídí tak, aby dodali v každém sprintu fungující SW

Extrémní programování (XP)

XP je velmi „lehkou“ metodikou pro malé až středně velké týmy (2–10 programátorů), které se musí vyrovnat s rychle se měnícím či nejasným zadáním při vývoji softwaru. Zaměřuje se na oblast softwarově-inženýrskou, řízení a organizaci. Jednoduchý návrh redukuje testovací práci. Neustálé testování redukuje čas dodávky. Prokládání kódování a testování dává vývojářům a testerům lepší porozumění kódu. Automatizované testy dovolují vývojářům provádět refaktorizaci.

XP používá známé principy a postupy, které dotahuje do extrémů:

- párové programování – neustálé revidování kódu,
- testování funkcionality
- refaktorizace

13. Řízení kvality při vývoji SW

Kvalita softwaru je shoda s explicitně stanovenými požadavky na funkci a chování, explicitně dokumentovanými vývojovými standardy a implicitními charakteristikami, které se očekávají od každého profesionálně vyvinutého softwaru.

Kvalitní software je takový software

- funguje v souladu s požadavky klienta
- odpovídá standardům
- vyvinutý s ohledem na obecně platné zásady tvorby konkrétního softwaru

Kvalita softwaru je ovlivněna mnoha faktory. Existuje mnoho modelů pro dělení těchto faktorů. Mezi ty nejznámější patří McCallův model, Deutsch-Willisův model, Evans-Marciniakův model, FURPS, jeho rozšíření FURPS+ a některé ISO standardy.

Zajištění kvality (software quality assurance – SQA)

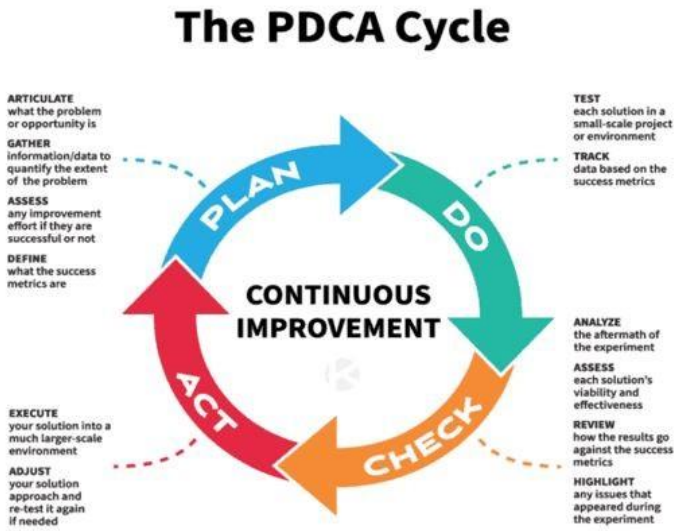
je souhrnný pojem, který zahrnuje již na začátku vývoje stanovení procesů a metod, jak správně **definovat požadavky na systém**, správně vyvíjet dílo a další dílčí etapy vývoje. Úlohou SQA je pochopitelně nejen stanovení procesů, ale také **kontrola procesů i všech výstupů**, kterou se zabývá řízení kvality (software quality control – SQC)

Kvalita a její zajištění a řízení jsou jedním z rozhodujících faktorů stabilního ekonomického růstu firem. Klíčovým pozitivním projevem dobře zavedeného systému řízení kvality je **rostoucí spokojenost a loajalita zákazníků**, **zefektivnění procesů kontroly, snížení nákladů a zvýšení produktivity**. Jelikož roste podíl napoprvé dobře provedené práce, snižuje se počet předělávání a oprav všeho druhu nebo dohledávání nejrůznějších dat a informací. Firmy s fungujícím systémem řízení jakosti dosahují dlouhodobě podstatně lepších výsledků než ostatní firmy Klíčová je snaha o neustálé zlepšování ve všech fázích vývoje, tak jak ji popisují cykly zlepšování PDCA.

PDCA

PDCA je metoda postupného zlepšování např. kvality výrobků, služeb, procesů, aplikací atd., probíhající iterativně ve čtyřech krocích.

- **plánuj (Plan)** – stanovování cílů dle stávajících podmínek, což zahrnuje určení úkolů, opatření a procesů k dosažení cílů,
- **dělej (Do)** – realizace cíle dle navrženého řešení,
- **kontroluj (Check)** – kontrola výsledku realizace oproti očekávaným výsledkům,
- **uskutečni (Act)** – vyhodnocení, na jehož konci se rozhodne, jestli měla úprava pozitivní efekt, pak se nový proces přidá do standardů. Pokud ne, tak se nebude nic měnit. Jestliže při vyhodnocení vyplynuly možnosti na zlepšení, tak se metoda vrací do prvního kroku a proces se opakuje.



Jádrem zajištění a řízení kvality je testování, mezi další techniky patří:

- **přezkoumání (revize)** – je činnost prováděná k zajištění vhodnosti, přiměřenosti, efektivnosti a účinnosti předmětu s cílem dosáhnout stanovených cílů,
- **inspekce** – je formální metoda přezkoumání podle přísných pravidel. Jejím cílem je především odkrýt chyby ve funkci, logice či implementaci, ověřit, že software vyhovuje požadavkům, ujistit se, že software je implementován podle definovaných standardů atd.,
- **audit** – je prověření dodržování a stavu plnění úkolů nezávislou skupinou. Některou specifickou oblast (např. účetní audit, audit kvality podle normy ISO 9000) může prověřovat jen akreditovaná organizace,
- **simulace** – z pohledu technik zajištění jakosti se jedná o ruční nebo poloautomatické procházení částí programu se symbolickým prováděním výpočtu.

Testování je **prověřování funkčnosti produktu**. Proces testování začíná **stanovením vize a cílů testování**. Dále se určí záběr testování, tedy co vše je třeba testovat, vybírají se testy, sbírají data a připravují nástroje, které tým k testování potřebuje. Navíc se kontroluje, zda všechny požadavky na produkt jsou v takové formě, aby bylo možno jednoznačně zkontrolovat jejich splnění. Samotné testování probíhá zkoumáním produktu na několika úrovních a reportováním nalezených skutečností. Proces testování se často provádí ve **více iteracích**, kdy každá iterace začíná předáním nové verze produktu testům.

Součástí zjišťování informací o kvalitě je **reportování nalezených softwarových chyb**. Mohli bychom předpokládat, že softwarové chyby vznikají na straně programátora, který něco přehlédl, nebo se mu v kódu objevila chyba. Není tomu

ale tak. Dle Rona Pattona je nejčastějším zdrojem chyb špatná specifikace. Je to způsobeno tím, že často dokumentace chybí úplně, je nedostačující nebo se v průběhu vývoje mění.

Metriky jsou způsoby a nástroje měření. Metrika vyjadřuje stav určitého systému, například jeho kvality, efektivnosti a nabývá při tom různých hodnot. Pro tento termín se používají také specializované pojmy **Performance indicator nebo Key performance indicator (KPI).**

U každého softwaru je potřeba specifikovat konkrétní metriky, které budou měřeny. Mezi webové metriky může patřit např. míra opuštění stránky, míra zájmu, míra konverze, čas strávený na stránce, podíl zobrazení ve vyhledávači atd. Nejčastěji jsou tyto metriky měřeny pomocí Google analytics

Kvalitu můžeme hodnotit z různých hledisek, a to z hlediska vyspělosti firmy (model CMMI) nebo hlediska systému kvality (normy).

Model CMMI

CMMI (Capability Maturity Model Integration) je model kvality organizace práce, který je určený pro vývojové týmy. Jde o souhrn cílů a doporučených pracovních postupů pro vývojové týmy, které vedou ke kvalitnímu plánování a řízení prací, a měly by zajistit i odpovídající kvalitu výstupu. CMMI definuje procesní oblasti, které musí tým realizovat, a cíle, kterých musí v každé oblasti dosahovat.

Standard rozděluje procesní oblasti do několika skupin podle typu činností:

- **řízení procesů** – zaměření se na procesy organizace, definice procesů organizace, školení organizace,
- **řízení projektů** – plánování, monitorování a řízení projektů, řízení vztahů se subdodavateli, řízení rizik, integrované řízení projektů,
- **návrh a realizace** – řízení požadavků, vývoj požadavků, technické řešení, integrace produktu, verifikace a validace,
- **podpůrné procesy** – řízení konfigurací, zajištění jakosti produktů a procesů, měření a analýza, rozhodování na základě analýzy variant.

Úrovně stupňovitého modelu

Stupňovitý model CMMI definuje 5 úrovní zralosti, přičemž model je navržen tak, aby firmy mohly kvalitu svých procesů přirozeně rozvíjet podle úrovní:

- počáteční (Initial),
- opakovatelný (Repeatable),
- definovaný (Defined),
- řízený (Managed),
- optimalizující (Optimizing)

U **počáteční úrovně dominují nahodilé (ad hoc) procesy**. Software je vytvářen bez firemních pravidel chaoticky, takže se jeho tvorba často dostává do kritických situací. Dosažený úspěch ve vývoji softwaru je důsledkem šťastné náhody a závisí na individuálních schopnostech a znalostech programátorů. Dohodnuté termíny nejsou většinou dodržovány a ukončení vývoje je dosahováno velkým úsilím na konci projektu. Celkové náklady na projekt jsou sečteny po ukončení vývoje a posuzují se jako důsledek vývoje a nutných výdajů. Kvalita softwaru není zajištěna.

Opakovatelná úroveň je **charakteristická opakovaným dosahováním dobrých výsledků**. Firma využívá základních postupů projektového řízení, ale z projektu na projekt se přenášejí jen některé úspěšné prvky řízení. Nicméně intuitivně **zaběhané procesy a povědomí, že je potřeba pracovat kvalitně**, vytvářejí dost stabilní prostředí pro udržení přijatelné úrovně jakosti softwarových produktů.

Pokud je společnost na třetí úrovni, tedy definované, **tak vyvíjí software podle předem stanoveného postupu**, metodicky, plánovitě, s využitím pokročilého projektového řízení s cílem dosáhnout vypracování požadovaného software v čase, s rozpočtovanými náklady a disponibilními zdroji. **Provádí se pravidelné vyhodnocování odchylek**

od plánu a přijímají se opatření ke krácení termínů jednotlivých činností, aby software yl dodán včas. **O kvalitu produktů a služeb se s ohledem na zákazníky explicitně usiluje**, proto je kvalita softwaru dodržována na velmi dobré úrovni a má tendenci vykazovat určité zlepšování.

Řízenou úroveň má firma, která má všechny procesy jasně definovány a je pro ně stanoven postup měření, který vyhodnocuje jejich podíl a efektivitu při tvorbě softwaru. **Zjištěné charakteristiky procesů jsou postupně upravovány tak, aby se firma přizpůsobila měnícím se podmínkám trhu**, aniž by to mělo dopad na jakost vyvíjeného softwaru, jehož kvalitativní parametry jsou firmou cílevědomě stále zvyšovány a dosahují vysoké úrovně.

Optimalizující úroveň je typická kontinuální zpětnou vazbou, která ovlivňuje následné softwarové projekty tak, aby se firemní procesy neustále zlepšovaly a dosahovaly předem definovaných, optimálních parametrů. **Firma dosahuje trvale špičkové kvality softwaru**, aniž by náklady na ni měly dopad na hospodaření firmy. Naopak, garantovaná kvalita softwaru usnadňuje jeho prodej.

Pro model byly definovány jednotlivé kroky, které umožňují postup na vyšší úroveň:

- z 1. na 2. úroveň – disciplína, zodpovědnost, pořádek, cílevědomost,
- z 2. na 3. úroveň – standardizace, stálost a provázanost firemních procesů, monitorování procesů,
- z 3. na 4. úroveň – kritéria a měření procesů, řízení změn a využívání predikce vývoje procesů,
- ze 4. na 5. úroveň – neustálé zlepšování procesů, vícekriteriální optimalizace procesů, adaptibilita procesů.

Důležité je si uvědomit, že množství rizika vývoje projektu se s rostoucí úrovní snižuje, na druhou stranu se zvyšuje kvalita projektu a produktivita týmu. Proto se firmy snaží zvyšovat svoji úroveň.

Řada norem **ISO 9000 definuje systém managementu jakosti**. Tyto normy vydává Mezinárodní organizace pro normalizaci. Normy umožňují prokázat daným organizacím schopnost výroby či distribuci produktů v souladu se všemi nezbytnými předpisy a potřebami zákazníka. ISO 9000 slučuje tři standardy ISO 9001, ISO 9002 a ISO 9003

Standard ISO 9001 slouží jako referenční model pro nastavení základních řídicích procesů v organizaci, které pomáhají neustále zlepšovat kvalitu poskytovaných výrobků a služeb, spokojenost zákazníka, strategické řízení a práci s riziky. Je to norma procesně orientovaná. Stejně jako ostatní normy **ISO vyžaduje následnou certifikaci zavedeného systému** řízení (zavedených procesů) v organizaci. Výsledkem je certifikát, který je mezinárodně uznávaný a je předpokladem určité zralosti a vyspělosti organizace.

Norma **ISO 9000 je určením blízká k modelu CMMI**, ale je mezi nimi několik zásadních rozdílů: Norma ISO 9000 není určena pro žádnou konkrétní oblast a je aplikována na firmy z nejrůznějších oborů. Naproti tomu CMMI je určena pro vývojové týmy. **ISO 9001 je stručný standard, který definuje pouze cíle**, zatímco CMMI je podrobný model, který jde do podrobností, když definuje očekávané činnosti a jejich pracovní výstupy. Díky tomu má CMMI návodný charakter, takže je na jeho základě možné procesy přímo definovat. V neposlední řadě **ISO 9001 nedefinuje stupně zralosti,** takže nevede firmy k soustavnému zlepšování tak jako CMMI

14. Analýza, návrh a realizace databáze

ER (Entity Relationship) modelování se používá pro abstraktní a konceptuální znázornění dat. Spočívá ve využití základních konstruktů jazyka pro tvorbu diagramů a v metodice tvorby těchto diagramů – základní myšlenkou je, že databáze uchovává fakta o entitách a o vztazích mezi entitami. Výsledkem je ERD (Entity Relationship Diagram, ER diagram).

Zásady

- Minimalizace redundance (normalizace)
- Maximalizovat znovupoužitelnost (dědičnost)
- Maximalizovat výkonnost (de-normalizace, metody, database tuning)
- Minimalizovat nároky na uložení dat (compression)

Využívá se přitom princpy tří architektur. Smyslem P3A je postupně upřesňovat datový model tak, aby zcela odpovídal požadavkům využívané databázové technologie. Stejně tak je naopak žádoucí, aby model v první vrstvě byl, pokud možno, abstraktní a tech. nezávislý.

Konceptuální – model reality

Logický – Popis způsobu realizace systému v termínech jisté třídy technologického prostředí (lineární, relační, hierarchické, síťové datové struktury) Např. u RDM – doplnění cizích klíčů.

Fyzický – Popis vlastní realizace systému v konkrétním implementačním prostředí (doplnění i typu indexu, velikosti, rozmístění pracovního prostoru apod.).

Využívá se taktéž metoda **normalizace dat**

Návrh databáze

Spočívá v určení entit (tabulek), klíčů (referenční integrita), vztahů (relace), dodržení normálních forem, indexů.

Entita

- tabulka (ale až v DB), která vykazuje stejné společné znaky
- něco, co je natolik důležité, že nám stojí zato to pojmenovat ⇒ v ER modelech modelujeme entitní typy, příklad:
- entitní typ: STUDENT, entita: Ondřej Horák
- výskyty entit musí být identifikovatelné na základě jejich atributů nebo vztahů s jinými entitami

Vztahy = Pojmenované spojitosti mezi entitami

Řádek = Kombinace sloupcových hodnot v tabulce. Obsahuje data vztahující se k jednomu objektu.

Sloupec = množina dat jednoho typu (jeden atribut)

- atribut je modelovaná vlastnost entit nebo vztahů zahrnutých do modelu
- každá vlastnost zjistitelná pro nějakou entitu (atribut) je v modelu zakreslena nejvýše jedenkrát
- existují také odvozené atributy, které se dají odvodit z vlastností jiných entit
- je několik typů atributu / sloupce
    - povinný
    - Volitelný – může nemít hodnotu
    - Vícehodnotový – má hodnoty definované v dané doméně (např. JménoOsoby— u SmluvníPartner)
    - Skupinový / složený– vnitřní struktura např. u adresy (ulice, město, psč) – obvykle se normalizuje
    - Atomický – není vícehodnotový ani skupiný

Vztahy

- mezi entitami lze popsat vztah větou, ve které vztah vyjádříme přísudkovou částí. Např. „Zákazník vytváří Objednávku“
- mohou být binární (vztahy dvou entit), ternární (vztah tří entit) i více-ární
- u vztahu zjišťujeme a do diagramu zakreslujeme tzv. kardinalitu vztahu
    - kardinalita vztahu říká, kolik výskytů entit jednoho typu může být v daném vztahu s jedinou entitou druhého typu – říkáme, že vztah je buď 1:1, nebo 1:N nebo N:1 nebo M:N
- dále zjišťujeme tzv. povinnost členství ve vztahu
    - říká, zda všechny výskyty entitního typu, jenž je určen pro danou roli v tomto vztahu, musí do tohoto vztahu skutečně vstupovat – například ne všichni učitelé musí v konkrétním semestru učit nějaký předmět
- někdy do konceptuálního modelu zahrnujeme i informaci o tom, zda je vztah přenositelný
    - například „Učitel učí Předmět“ může být přenositelný vztah, kdy se učitelé ve výuce daného předmětu střídají semestr po semestru.
- Zpracování vztahů

- 1:1 – záznam obsahuje sloupec pro zapsání právě jednoho primárního klíče jiného záznamu
- 1:M – záznam obsahuje sloupec pro zapsání právě jednoho primárního klíče jiného záznamu – na onen jiný záznam se může odkazovat více různých záznamů
- M:N – může existovat více vzájemných odkazů – realizováno pomocí samostatné vazební tabulky (=dekompozice, neboli rozdělení na dvojici vztahů 1:M)

Integrita = pravidla pro zajištění správnosti a konzistence

- **Entitní** = nutnost existence primárního klíče pro každý záznam
- **Referenční** = odkazování pouze na exitující záznamy v povolené kardinalitě
- **Doménová** = stanovení povolených hodnot ve sloupci

Indexy = databázový objekt, sloužící ke zrychlení vyhledávacích a dotazovacích procesů v databázi, definování unikátní hodnoty sloupce tabulky nebo optimalizaci fulltextového vyhledávání.

- Každý index zabírá v paměti vyhrazené pro databázi nezanedbatelné množství místa (vzhledem k paměti vyhrazené pro tabulku). Při existenci mnoha indexů se může stát, že paměť zabraná pro jejich chod je skoro stejně velká, jako paměť zabraná jejími daty – zvláště u rozsáhlých tabulek (typu faktových tabulek v datovém skladu) může něco takového být nepřijatelné.
- Každý index zpomaluje operace, které mění obsah indexovaných sloupců (INSERT, UPDATE). To je dáno tím, že databáze se v případě takové operace nad indexovaným sloupcem musí postarat nejen o změny v datech tabulky, ale i o změny v datech indexu.
- Pro správné zvolení indexů by ten, kdo databázi navrhuje, měl vědět, jak často se vybírané záznamy z dané tabulky třídí podle zamýšleného sloupce (a kandidáta na index) a nakolik je důležité, aby vyhledávání a třídění podle něj bylo rychlé. U některých tabulek, které jsou např. číselníky o stálém počtu položek, řekněme max. několika desítkách, nemusí být třeba index definován vůbec.

15. Datové modelování

Při datovém modelování na základě znalostí o realitě navrhujeme struktury dat, do kterých se budou o této realitě ukládat záznamy. Je to činnost, která vyžaduje důkladnou analýzu skutečnosti, a schopnost převést získané představy do databázových struktur. V souhrnu se této činnosti říká návrh databáze. Analýza informací o skutečnosti – je náročná, a vyžaduje zkušenosti, jasný rozum, někdy dovednost obratně se ptát lidí, kteří danou skutečnost znají.

Datové modelování – činnosti

- Rozlišení množin objektů (entitních množin)
- Pojmenování entitních množin a identifikace entit
- Rozlišení entitních podmnožin (zaměstnanec – učitel, vědec, administrátor, zaměstnanec)
- Určení vztahů mezi entitními množinami, určení kardinality (počet) a parciality (volitelnost) vztahů
- Určení atributů entitních množin a vztahů
- Vyřešení problémů synonym a homonym

Data v tabulce jsou uložena v polích uspořádaných do řádků (= záznamy) a sloupců (= atributy)

Transformace do relačního modelu dat

- Entity → tabulky
- Atributy → sloupce
- 1:N vztahy → FK
- M:N vztahy → asociativní tabulky
- identifikátory → PK
- volitelnost → (povolené hodnoty null FK)
- povinnost → (not null FK)

Pravidla transformace konceptuálního schématu do relačního modelu dat

- Entitní množina → 1 tabulka, identifikátory => atributy tvořící PK tabulky
- Vztah je vyjádřen cizím klíčem
    - Jaká varianta je vhodnější pro transformaci vztahu 1:1?
- Vztah M:N je vyjádřen vazební relační tabulkou s dvěma FK
- Generalizace / specializace – více variant
    - 1 relační tabulka
    - Relačními tabulkami na úrovni specializovaných entitních množin
    - Relačními tabulkami na úrovni generalizující entitní množiny i na úrovní specializovaných entitních množin

Základní pojmy

- entita: typ objektů, ať už konkrétních (osoby, místa, věci) či abstraktních (katalogové položky, kategorie), natolik důležitých v dané realitě, že se o takových objektech mají vést záznamy
- objekt: jiné označení pro entitu
- atribut: vlastnost nebo charakteristika objektu nebo vztahu v datovém modelu (např. jméno a příjmení zaměstnance)
- vztah: pojmenovaná spojitost mezi entitami
- kardinalita: četnost vazby; nabývá hodnot 1:1, 1: N, M:N
- parcialita: volitelnost vztahu mezi entitami

Principy datového modelování

První část – analýza informací o skutečnosti

- je náročná, a vyžaduje zkušenosti, jasný rozum, někdy dovednost obratně se ptát lidí, kteří danou skutečnost znají
- v první fázi se vytváří konceptuální schéma a poté konceptuální datový model
- viz otázka 32. Konceptuální modelování obsahu databáze

Druhá část – převod do databázových struktur

- následuje převod na fyzický datový model pro konkrétní databázový systém
- rozložení složených atributů, přiřazení datových typů, výběr Primary Key, rozhodnutí o realizaci dědičnosti, vyřešení vztahů 1:1, vztahy m:n do samostatné tabulky, samostatné entitní typy do samostatné tabulky
- normalizace (normální formy)

Komponenty

- K zachycení datového modelu na konceptuální úrovni se používá řada různých modelovacích nástrojů. Mezi nejznámější patří různé modifikace tzv. ER(A) diagramů.

Entita

- Rozlišitelný a identifikovatelné objekt reality. Entitou je např. Karel Novák, katedra informačních technologií, motor Š-135-12333.
- Entity se slučují do entitních množin (typů). Každá množina musí mít uveden identifikátor (tj. minimální množinu atributů, které zajišťují jednoznačnou identifikaci entit v této množině)

Rozlišují se následující typy entit:

- Obecná
- Silná/kmenová/základní/regulární (existují nezávisle na jiných entitách)
- Slabá/popisná
- Vazební/asociativní (realizuje vazbu mezi entitami)

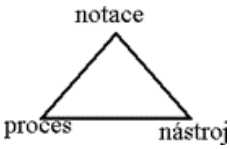
- Generalizace/nadtyp (vytvoření entity vyšší úrovně z dvou a více entit nižší úrovně)
- Specializace/podtyp

Notace

- hraje důležitou roli v každém modelu
- způsob interpretace obrázků/modelů a nástroj, který zaručuje konzistentní proces vývoje

Trojúhelník úspěšnosti

- Pro úspěšný projekt je třeba mít tři předpoklady – notaci, proces a nástroj.
- Máme-li notaci, ale nevíme, jak ji použít (proces), budeme neúspěšní. Máme-li silný proces, který ale neumí komunikovat s notací, budeme neúspěšní. A jestliže nemáme možnost dokumentovat výsledek naší práce (nástroj), pravděpodobně budeme opět neúspěšní.



UML

- Více o UML viz otázku 10. Modelovací jazyk UML

BPMN

- Zkratka od Business Process Modeling Notation. Jedná se o grafickou notaci (soubor grafických objektů a pravidel pro jejich spojování) určenou pro modelování procesů. Definuje pouze jediný diagram – BPD (Business Process Diagram).

ER (Entity Relationship) modelování se používá pro abstraktní a konceptuální znázornění dat. Spočívá ve využití základních konstruktů jazyka pro tvorbu diagramů a v metodice tvorby těchto diagramů – základní myšlenkou je, že databáze uchovává fakta o entitách a o vztazích mezi entitami. Výsledkem je ERD (Entity Relationship Diagram, ER diagram).

Zásady

- Minimalizace redundance (normalizace)
- Maximalizovat znovupoužitelnost (dědičnost)
- Maximalizovat výkonnost (de-normalizace, metody, database tuning)
- Minimalizovat nároky na uložení dat (compression)

P3A – princip tří architektur

Smyslem P3A je postupně upřesňovat datový model tak, aby zcela odpovídal požadavkům využívané databázové technologie. Stejně tak je naopak žádoucí, aby model v první vrstvě byl, pokud možno, abstraktní a tech. nezávislý.

Konceptuální – model reality

- Model obsahu datové základny na konceptuální úrovni (tj. nezatížený jakýmikoliv implementačními a technologickými implementacemi)
    - KSR – model obsahující základní entitní množiny, vztahy a jejich atributy. Jedná se o hrubý model vytvořený za účelem poznání zkoumané reality.
    - KSD – popis obsahu datové základny za účelem přesné specifikace obsahu datové základny. Předběžný návrh datové základny

Logický – Popis způsobu realizace systému v termínech jisté třídy technologického prostředí (lineární, relační, hierarchické, síťové datové struktury) Např. u RDM – doplnění cizích klíčů. Při transformaci do relační se řídí pravidly:

- Každá entitní množina je transformována do jedné relační tabulky. Identifikátory entitní množiny se stanou atributy tvořícími primární klíč relační tabulky.
- Vztah z konceptuálního modelu je v relačním modelu vyjádřen cizím klíčem
- Vztah M:N z konceptuálního modelu je v relačním modelu vyjádřen vazební relační tabulkou a dvěma cizími klíči.

Generalizace / specializace je transformována do relačního modelu dat několika variantami:

- jednou relační tabulkou na úrovni celé hierarchie,
- relačními tabulkami na úrovni specializovaných entitních množin,
- relačními tabulkami na úrovni generalizující entitní množiny i na úrovni specializovaných entitních množin.

Využívá se taktéž metoda **normalizace dat**

- Cíle:
    - vytvořit co nejvěrnější obraz v modelovaném světě existujících entitních množin
    - zajistit interní konzistenci datového modelu, resp. databáze,
    - minimalizovat redundance
    - maximalizovat stabilitu datových struktur
- Sada normálních forem (první, druhá, třetí, boyce/codd, čtvrtá, pátá).

Fyzický – Popis vlastní realizace systému v konkrétním implementačním prostředí (doplnění i typu indexu, velikosti, rozmístění pracovního prostoru apod.). Je dán:

- Způsobem uložení dat
    - Sekvenční = Sekvenční způsob uložení dat není využíván pro uložení dat v databázových systémech. Bývá používán pro zálohování databázových souborů.
    - Přímé = hodnotě primárního klíče jednotlivých záznamů je přiřazena (vypočtena) určitým algoritmem (hashing algorithm) adresa uložení záznamu na paměťovém médiu
- Způsobem přístupu k datům
    - Sekvenční
    - Přímé
    - S využitím dalších mechanismů (řetězení, indexy)

16. Databázové jazyky

Základní pojmy

- data: formalizované a fyzicky zaznamenané znalosti, poznatky, zkušenosti, výsledky pozorování
- informace: smysluplná interpretace dat
- databáze: integrovaná počítačově zpracovaná množina persistentních dat
- relační model dat: založený na predikátové logice prvního řádu, k manipulaci s daty možno použít relační kalkul nebo operace relační algebry
- dotazovací jazyk: slouží pro získání dat z databáze; v relační databázi je odpovědí relace, jejíž n-tice splňují nadefinované podmínky
- systém řízení bází dat (SŘBD): množina programovým prostředků, která umožňuje:
    - vytvoření databáze
    - použití databáze (manipulace s daty – výběr, vkládání, update, mazání)
    - údržbu a správu databáze
- databázový systém (DBS): SŘBD + databáze

Význam a užití

- Prostředek komunikace člověka s databázovým systémem
- **Pravidlo komplexního datového jazyka:** Relační systémy mohou podporovat více jazyků a režimů přístupů, ale musí existovat minimálně jeden jazyk, jehož příkazy jsou vyjádřitelné dobře definovanou syntaxí jako řetězce znaků, který podporuje definici dat, manipulaci s daty, ochranu dat a omezení integrity (lze považovat za standard)

Základní dělení jazyků podle nejčastěji identifikovaných částí

- **DDL (Data Definition Language):** definice databázových objektů; příkazy create, alter, drop

- **DML (Data Manipulation Language):** manipulace s daty; příkazy select, insert, update, delete
- **DCL (Data Control Language**): řízení přístupu k datům; příkazy grant, revoke
- **TCL (Transaction Control Language):** řízení transakcí; příkazy commit, rollback, savepoint

Příklady využití

- Databáze jako samostatný prvek – využití pro evidenci a správu dat
- Databáze jako podpora aplikace – uložení dat do databázové struktury, ze které aplikace čerpá

Příklady dotazovacích databázových jazyků 4. generace

Databázové jazyky 4. generace (4GL: 4th Generation Language; pokročilé programovací jazyky, které dovolí laikům (neprogramátorům) psát krátké programy extrahující data z databází a vytvářet reporty).

- FOCUS
- Informix-4GL
- Quel
- Progress 4GL
- SQL (v současné době nejrozšířenější)

Databázový jazyk SQL

Standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. SQL je zkratka anglických slov **Structured Query Language** (strukturovaný dotazovací jazyk).

V 70. letech 20. století probíhal ve firmě **IBM** výzkum relačních databází. Bylo nutné vytvořit sadu příkazů pro ovládání těchto databází. Vznikl tak jazyk **SEQUEL (Structured English Query Language**). Cílem bylo vytvořit jazyk, ve kterém by se příkazy tvořily syntakticky co nejblíže přirozenému jazyku (angličtině). Relační databáze byly stále významnější, a bylo nutné jejich jazyk standardizovat. Americký institut ANSI původně chtěl vydat jako standard zcela nový jazyk RDL. SQL se však prosadil jako de facto standard a ANSI založil nový standard na tomto jazyku. Tento standard bývá označován jako SQL-86 podle roku, kdy byl přijat.

Standardy podporuje prakticky každá relační databáze, ale obvykle nejsou implementovány vždy všechny požadavky normy. A naopak, každá z nich obsahuje prvky a konstrukce, které nejsou ve standardech obsaženy. Přenositelnost SQL dotazů mezi jednotlivými databázemi je proto omezená.

Postupně standardizován organizacemi:

- **ISO**: Mezinárodní organizace pro normalizaci (International Organization for Standardization)
- **IEC**: Mezinárodní elektrotechnická komise (International Electrotechnical Commision)
- **ANS**I: Americká standardizační organizace (American National Standards Institute)

Vývoj standardů:

Rok	Označení	Charakteristika
1986	SQL-86	První verze (ANSI / ISO 1987)
1989	SQL-89	Oprava a mírné rozšíření normy jazyka SQL-86
1992	SQL-92	Označení SQL2, komplexní pohled na jazyk, tři úrovně
1999	SQL:1999	Nepřesně označováno jako SQL3. Výrazné rozšíření jazyka, regulární výrazy, rekurzivní dotazy, triggery, objektově orientované vlastnosti
2003	SQL:2003	Podpora XML, GUI funkce, multimédia
2006	SQL:2006	Import, export a způsob použití XML dat, vazba na XML Query Language (W3C)
2008	SQL:2008	Kompletní přepracování všech hlavních částí jazyka SQL

Minimální požadavky dle platných standardů databázového jazyka SQL:

ISO/IEC 9075-1:2008	Information technology – Database languages – SQL – Part 1: Framework (SQL/Framework)
ISO/IEC 9075-2:2008	Information technology – Database languages – SQL – Part 2: Foundation (SQL/Foundation)
ISO/IEC 9075-11:2008	Information technology – Database languages – SQL – Part 11: Information and Definition Schemas (SQL/Schemata)

I přes přesný popis standardů přežívá (z důvodu zpětné kompatibility) řada odchylek – **dialektů** – implementací jazyka SQL výrobci jednotlivých databázových systémů.

Příklady použití SQL

Vytvoření tabulky

CREATE TABLE název tabulky (atribut, typ, PK, NOT NULL...);

CREATE TABLE zamestnanci (ID_zam int primary key, jmeno_zam varchar (20) NOT NULL...);

Smazání tabulky

DROP TABLE název tabulky

DROP TABLE zamestnanci

Vložení dat

INSERT INTO název tabulky (název atributu) VALUES (hodnota atributu);

INSERT INTO zamestnanci (ID_zam, jmeno_zam) VALUES (1, 'Novák');

Výběr dat

*SELECT * FROM název tabulky;*

*SELECT * FROM zamestnanci;*

Aktualizace dat

UPDATE název tabulky SET název atributu = nová hodnota WHERE podmínka;

UPDATE zamestanci SET jmeno_zam = Nováková WHERE ID_zam = 1;

Mazání dat

DELETE FROM název tabulky WHERE podmínka;

DELETE FROM zamestnanci WHERE jmeno_zam='Novák';

PL/SQL

Procedural Language / Structured Query Language je procedurální nadstavba jazyka SQL od firmy Oracle založená na programovacím jazyku Ada.

Pomocí PL/SQL je možno vytvářet:

- Uložené procedury a funkce
- Programové balíky
- Triggery
- Uživatelsky definované datové typy

Tato nadstavba se rozšířila a její deriváty převzaly i jiné relační databáze.

OQL

Object Query Language (OQL) je **standardizovaný dotazovací jazyk pro objektově-orientované databáze** vytvořený podle SQL. OQL byl vyvinut Object Data Management Group (ODMG). Kvůli jeho komplexitě **zatím žádný výrobce neimplementoval OQL v úplnosti**. OQL ovlivnil návrh některých novějších dotazovacích jazyků jako JDOQL a EJB QL. Jazyk OQL je záměrně navržen tak, aby byl velmi podobný jazyku SQL-92. Dokonce platí, že část syntaxe SQL, která se týká manipulace s daty, je kompletně obsažena v OQL. Proto je například řada dotazů konstrukce select-from-where zcela shodná v SQL i OQL.

Na druhou stranu se jazyk OQL nezabývá tvorbou datových struktur. To znamená, že příkazy SQL týkající se definice dat (např. create-table, create-index atd.) nemají v OQL žádný ekvivalent. K definici tříd, kolekcí a metod jazyk OQL nejde použít. Počítá se zde buď s nějakým běžným objektovým programovacím jazykem jako např. Java, C#, C++, Smalltalk

Databázovým dotazem je nejen konstrukce select-from-where, ale každý výraz, který reprezentuje nějaká data. Proto například prostý výraz v OQL zaměstanci se v SQL musí napsat jako select * from zaměstnanci

JPQL

Java Persistence Query Language je platformě nezávislý objektově-orientovaný dotazovací jazyk definovaný jako součást specifikace Java Persistence API (JPA). JPQL se používá k dotazování se na entity uložené v relační databázi. Je silně inspirován SQL a má podobnou syntaxi. JPQL odstiňuje programátora od specifik konkrétního datového úložiště. Jinými slovy, programátor se nestará o to, zda jsou v konečném důsledku dotazy prováděny nad Oracle nebo MySQL databází. Stále píše stejné dotazy. Tato vlastnost je velice pohodlná, může však mít i jistá omezení. Každý výrobce databází přináší do svého produktu jinou funkcionalitu, kterou se odlišuje od ostatních.

Používáním standardních „univerzálních“ dotazů pomocí JQPL o tyto výhody přichází. Někdy se v této souvislosti hovoří o zákonu netěsných abstrakcí. Ten ve zkratce říká, že jakákoli abstrakce vede k zúžení funkcionality nebo k neefektivnímu využívání zdrojů. Jako příklad se často uvádí tato abstrakce SQL:

- WHERE a=b AND b=c
- WHERE a=b AND b=c AND a=c

Matematicky jsou tyto zápisy totožné. Může však u některých databází docházet v případě použití druhého zápisu k výraznému zlepšení výkonnosti zpracování.

XQuery

XQuery je dotazovací a **funkcionální programovací jazyk** navržený pro dotazování na kolekce dat v XML. XQuery poskytuje prostředky pro extrahování a manipulaci dat z XML dokumentů nebo jakéhokoli datového zdroje, který může být jako XML zobrazen, například relační databáze nebo některé dokumenty (např. OpenOffice, MS Office 2007,…).

XQuery používá **syntax jazyka XPath** pro adresování konkrétních částí XML dokumentů. To je doplněno SQL-podobnými „**FLWOR výrazy**“ pro provádění Joinů. FLWOR výraz je sestaven z pěti klauzulí, po kterých je pojmenován: FOR, LET, WHERE, ORDER BY, RETURN.

17. Přidaná hodnota dat pro business

Společnosti uchovávají velké množství dat ze systémů, které používají (ERP, CRM, SCM…). Tato data jsou zpravidla uložena v transakčních databázích (OLTP– OnLine Transaction Processing databáze). Ty jsou navrženy tak, aby umožňovaly ukládání velkého množství dat, jejich snadnou úpravu a mazání. Pokud ale dojde na získávání souhrnných informací, jsou transakční databáze pro tento účel naprosto nevhodné. Dalším problémem je, že analytická data, která potřebuje management firmy, pocházejí často z různých systémů a jsou tedy uloženy v různých databázích s odlišnou strukturou, což činí jejich získání ještě obtížnějším.

Technologie BI dokáží integrovat data ze všech oblastí podnikové činnosti a poskytnout potřebné informace v požadované struktuře. Ze získaných údajů se zpracovávají nové údaje, které nejsou zřejmé na první pohled. Ty jsou poté v čitelné podobě reprezentované manažerům, kteří na základě nich řídí své strategické rozhodování. Na rozdíl od ostatních komponent ERP je BI určen pro vrcholový management. BI pracuje s daty v reálném čase – na základě dotazu dostane uživatel odpověď během několika vteřin.

Přidaná hodnota dat pro business se stala jedním z nejdůležitějších faktorů úspěchu v dnešní digitální době. Využívání dat a analýza přináší organizacím možnost získat hlubší porozumění jejich operacím, zákazníkům a tržním trendům. Tyto informace jsou poté využívány k lepšímu rozhodování, zlepšení efektivity a inovacím, což vede k zvýšení konkurenceschopnosti a růstu.

- **Zlepšené rozhodování:** Analytické nástroje a techniky umožňují organizacím rozhodovat na základě dat a faktů. Analytické modely a algoritmy mohou odhalovat zákonitosti a vzory v datech, což umožňuje předvídat budoucí události a trendy a adekvátně na ně reagovat.
- **Optimalizace procesů:** Analytické nástroje a techniky jsou využívány k identifikaci slabých míst a neefektivních postupů v procesech organizace. Tyto poznatky umožňují organizacím optimalizovat své procesy a zlepšit efektivitu a produktivitu.
- **Lepší porozumění zákazníkům:** Data a analýza umožňují organizacím lepší pochopení chování a preferencí svých zákazníků. Toto porozumění umožňuje organizacím lépe cílit své marketingové kampaně, personalizovat své produkty a služby a budovat silnější vztahy se zákazníky.
- **Inovace a konkurenční výhoda:** Analytické nástroje mohou být použity k identifikaci nových příležitostí na trhu a inovacím v produktech a službách. Organizace, které efektivně využívají data a analýzu, mohou získat konkurenční výhodu a posílit svou pozici na trhu.
- **Predikce a prevence rizik:** Analytické nástroje mohou být použity k předpovídání budoucích událostí a identifikaci potenciálních rizik a problémů. Toto umožňuje organizacím přijímat preventivní opatření a minimalizovat negativní dopady.
- **Zlepšená interní a externí komunikace:** Analytické nástroje mohou být využity k zlepšení interních komunikačních procesů v organizaci a poskytování lepších informací a služeb zákazníkům.

Celkově vzato, využívání dat a analýza přináší organizacím mnoho příležitostí k zlepšení výkonnosti, inovacím a konkurenceschopnosti. Organizace, které efektivně využívají data a analýzu, mají možnost dosáhnout strategické výhody v rámci svého odvětví a posílit svou pozici na trhu.

18. Data, kvalita dat (základní pojmy a jejich důležitost)

Data jsou základním stavebním kamenem moderního informačního věku. Jsou to surové informace, které mohou být zpracovány a analyzovány pro získání užitečných znalostí a podpory rozhodovacích procesů. Kvalita dat ovlivňuje spolehlivost analýz, efektivitu rozhodování a konečné výsledky organizace.

Pojmy:

- **Data Analysis:** Jedná se o proces analyzování dat s cílem odhalit užitečné informace, vzory nebo znalosti. Zahrnuje různé techniky a metody interpretace výsledků.
- **Data Analytics:** Aplikace počítačových systémů na analýzu velkých datasetů s cílem podporovat rozhodování v organizaci. Data Analytics se zaměřuje na využití dat k odhalení trendů a predikci budoucího chování.
- **Data Science**: Interdisciplinární obor, který kombinuje statistiku, matematiku, strojové učení a další techniky k analýze a porozumění datům.
- **Data Mining:** Proces objevování vzorů nebo znalostí z velkých datasetů pomocí různých metod, jako jsou klastrování, klasifikace nebo asociační pravidla.
- **Machine Learning:** Metoda v oblasti umělé inteligence, která umožňuje počítačům „učit se“ z dat a vytvářet modely a predikce bez explicitního programování.
- **Artificial Intelligence:** Vědecký obor zabývající se vývojem inteligentních agentů, kteří jsou schopni vnímat své prostředí a přijímat kroky ke splnění stanovených cílů.

Datové zdroje:

- **Interní data:** Souhrn dat, která jsou generována nebo shromažďována v rámci organizace. Tato data zahrnují informace o transakcích, zákaznících, zaměstnancích a provozních procesech.
- **Externí data:** Data, ke kterým lze získat přístup mimo organizaci. Mohou to být veřejně dostupná data, jako jsou otevřená data vládních agentur nebo data z partnerských firem.
- **Strukturovaná data:** Data, která jsou organizována do tabulek nebo relačních databází a mají pevně definovanou strukturu a schéma.
- **Semi-strukturovaná data:** Data, která mají částečnou strukturu, jako jsou XML nebo JSON soubory, ale nejsou plně organizována do tabulek.
- **Nestrukturovaná data:** Data, která nejsou organizována nebo strukturována v žádném formátu a mohou obsahovat textové dokumenty, obrázky, zvuky apod.

Důležitost datové kvality:

Nástroje pro zajištění datové kvality se zaměřují na různé aspekty, aby zajistily, že data jsou použitelná, spolehlivá a relevantní:

- **Úplnost:** Zajištění, že data jsou kompletní a nejsou chybějící nebo nepoužitelná.
- **Soulad:** Ujistění se, že data jsou uložena ve standardním formátu a jsou srozumitelná a použitelná.
- **Konzistence:** Zajištění, že data jsou konzistentní a nejsou v konfliktu s jinými daty.
- **Přesnost:** Zajištění, že data jsou přesná a aktuální a odpovídají skutečnosti.
- **Unikátnost:** Identifikace a odstranění duplicitních záznamů.
- **Integrita:** Zajištění, že data mají správné vztahy a jsou logicky propojena.

Metody zajištění datové kvality:

- **Automatizace procesů:** Využití automatizace pro kontrolu a validaci datových procesů zvyšuje konzistenci a efektivitu.
- **Umělá inteligence:** Technologie umělé inteligence, jako jsou strojové učení a analýza vzorů, mohou identifikovat a řešit problémy s datovou kvalitou.
- **Správa metadat:** Kvalitní správa metadat pomáhá zajistit srovnatelnost a interoperabilitu dat mezi různými systémy a organizacemi.

Automatizace procesů a využití umělé inteligence pro identifikaci a řešení problémů s datovou kvalitou jsou stále více rozšiřované trendy v moderním informačním prostředí.

Příčiny nekvalitních dat:

- **Chyby při zadávání dat**: Nekvalitní data mohou být způsobena lidskými chybami při ručním zadávání informací.
- **Nesoulad mezi zdroji dat:** Data z různých zdrojů mohou obsahovat rozpory nebo nesouladné informace.
- **Zastaralá data:** Data, která nejsou aktualizována, mohou být zastaralá a nepřesná.
- **Nedostatečná dokumentace:** Nedostatečná dokumentace datových procesů a pravidel může vést k nekvalitním datům.

Kvalita dat je klíčová pro úspěšné rozhodování, plánování a operace v organizaci. Bez kvalitních dat jsou analýzy neúplné a nepřesné, což může vést k chybným rozhodnutím a ztrátě konkurenční výhody. Z tohoto důvodu je důležité klást důraz na správu dat a zajištění jejich kvality jako součást strategie informačního managementu organizace.

19. Agregovaná (odvozená) data, dimenze, metriky/KPI, granularita

Agregovaná (odvozená) data:

- **Definice:** Agregovaná (odvozená) data jsou zpracovanou formou původních údajů, která jsou seskupena nebo transformována za účelem získání souhrnných informací. Tento proces agregace může zahrnovat výpočet součtů, průměrů, počtů nebo jiných statistik z původních dat.
- **Příklad:** Při agregaci denních prodejních údajů za měsíc získáme celkový měsíční obrat. Typy agregovaných dat zahrnují sumarizované (např. součty, průměry) a derivované (např. procentuální změny).

Dimenze:

- **Definice:** Dimenze jsou kategorie, které popisují různé aspekty dat a umožňují organizaci dat do hierarchické struktury. Tyto kategorie mohou zahrnovat časová období, geografické lokality, produkty, zákazníky nebo jiné relevantní charakteristiky.
- **Příklad:** Pro prodejní údaje mohou být dimenze časové období (denní, týdenní, měsíční), geografické lokality (země, regiony), produkty (kategorie, typy) nebo zákazníci (segmenty, preference).

Metriky/KPI (Key Performance Indicators):

- **Definice:** Metriky/KPI jsou měřitelné ukazatele, které slouží k hodnocení výkonnosti organizace nebo určitého procesu. Tyto ukazatele jsou často definovány v souladu s cíli organizace a umožňují sledovat dosahování těchto cílů.
- **Příklad:** Pro prodejní údaje mohou metriky/KPI zahrnovat celkový obrat, ziskovost (hrubý a čistý zisk), počet nových zákazníků, míru konverze nebo průměrnou hodnotu objednávky.

Granularita:

- **Definice:** Granularita se týká úrovně detailu nebo jemnosti dat. Vyšší granularita znamená detailnější údaje, zatímco nižší granularita představuje údaje ve seskupené nebo souhrnné formě.
- **Příklad:** Pro prodejní údaje může být granularita určena úrovní detailu, jako jsou denní, týdenní, měsíční nebo roční údaje. Může se také lišit podle úrovně detailu v jednotlivých dimenzích, například granularita časových údajů může být denní, zatímco geografických údajů může být regionální.

Vztahy mezi pojmy:

- Agregovaná data jsou často odvozena z původních údajů a analyzována pomocí metrik/KPI.
- Dimenze poskytují kontext pro metriky/KPI a umožňují uživatelům zkoumat údaje z různých perspektiv.
- Granularita ovlivňuje detailnost a úroveň informací v agregovaných údajích a metrikách/KPI.

BSC

Balanced Scorecard (BSC) je strategický systém měření výkonnosti podniku. Jedná se o metodu vytvořenou tvůrci R. S. Kaplanem a D. P. Nortonem, která vznikla na základě výzkumné studie o nových metodách měření výkonnosti podniku. Výsledek této studie byl poprvé prezentován v roce 1992. Na základě této studie vyšla kniha The Balanced Scorecard (1996), která byla po prvé přeložena do českého jazyka v roce 2000.

Základní myšlenka

BSC vychází z vize a strategie podniku. BSC je souborem měřítek, která jsou odvozená od strategie organizace. Výkonnost podniku měří ze 4 úhlů pohledu (perspektiv), kterými jsou:

- **Finance**
- **Zákazník**
- **Interní podnikové procesy**
- **Učení se a růst**

Každá oblast obsahuje strategický záměr, cíle, metriky, iniciativy (akce). Základní myšlenkou je soustředit organizaci na měřítka, která jsou klíčová při naplňování strategie a dosahování strategických cílů. Implementace BSC znamená převedení strategie do konkrétních akcí, tj. převedení obsahu vize, strategie do jasných a měřitelných cílů v rámci výše uvedených perspektiv.

BSC představuje/poskytuje strategický rámec pro:

- Měření výkonnosti podniku
- Sdílení a převedení podnikových vizí
- Strategie skrz všechny úrovně řízení až k řadovým zaměstnancům

Cíle BSC:

- Efektivní management založený na metrikách
- Prosazení strategických cílů v celé organizaci
- Vyžadujeme od společností komplexní systémy, které dokážou nejen zpětně hodnotit výkonnost, ale také měřit potenciál do budoucích období

Přínosy BSC:

- Lepší přenos strategie do operativy (dříve se podniky potýkaly s problémem zrealizovat podnikovou vizi)
- Tvorba strategie se stala kontinuálním procesem
- Sladění jednotlivých procesů, služeb, kompetencí a jednotek organice

4 dimenze:

- **Finance** (uspokojování zájmů vlastníků společnosti, ukazatelé: ROI, ROCE, EVA, Cash Flow)
- **Zákazník** (spokojenost zákazníka, ukazatelé: počet a věrnost zákazníků, počet nových, podíl na trhu)
- **Interní podnikové procesy** (měření ukazatelů kvality výrobku a služeb, stav a rozvoj klíčových procesů)
- **Učení se a růst (**zaměstnanci – schopnosti, schopnost inovací)

Pro každou dimenzi určit mise, cíle, metriky, iniciativy (akce), očekávané hodnoty (targets), termín dokončení a odpovědná osoba za provedení akce.

20. Strojové učení a data mining

Data mining – dolování dat

- umožňuje pomocí speciálních algoritmů automaticky objevovat v datech strategické informace
- analytická technika pevně spjatá s datovými sklady, jako s velmi kvalitním datovým zdroje pro tyto speciální analýzy
- proces extrakce relevantních, předem neznámých nebo nedefinovaných informací z velmi rozsáhlých databází
- slouží manažerům k objevování nových skutečností, čímž pomáhají zaměřit jejich pozornost na podstatné faktory podnikání, umožňují testovat hypotézy, odhalují ve stále se zrychlujícím a složitějším obchodním prostředí skryté korelace mezi ekonomickými proměnnými apod.
- cílem je poskytovat strategické informace širokému aspektu manažerů v organizaci
- je založeno na množství matematický a statistických technik, např:
    - rozhodovací stromy,
    - neuronové sítě,
    - genetické algoritmy,
    - clustering a klasifikace
- zaměření na odlišné uživatele – dolování dat je prováděno automaticky, podle určených algoritmů a tak jejich cílovým uživatelem může být i manažer bez speciálních znalostí statistiky

Nástroje pro řízení kvality dat a správu metadat

Vzhledem k povaze řešení – podpoře analytické práce – je důležité, aby tato práce probíhala nad korektními daty, dokumentujícími přesnou situaci podniku. Nástroje pro zajištění **datové kvality** se proto zabývají zpracováním dat s cílem zajistit jejich:

- **Úplnost** – identifikována a ošetřena chybějící nebo nepoužitelná data
- **Soulad** – identifikována a ošetřena data, která nejsou uložena ve standardním formátu
- **Konzistenci** – identifikována a ošetřena data, jejichž hodnoty jsou v konfliktu s jinými
- **Přesnost** – identifikována a ošetřena data, která jsou nepřesná nebo zastaralá
- **Unikátnost** – identifikovány a ošetřeny duplicitní záznamy
- **Integritu** – identifikována a ošetřena data, která postrádají vztahy vůči ostatním datům

Nástroje pro správu metadat získaly na důležitosti až s implementací BI**. Metadata** jsou reprezentována jako **data o datech** a jsou **popisem obsahu a funkcionality veškerých informačních systémů i jejich jednotlivých částí**. Důvodem pro existenci metadat je nutnost popsat obsah a principy fungování jednotlivých komponent jakéhokoli IS/ICT řešení. Z pohledu BI zahrnují především datové modely, popisy funkcí, business a transformačních pravidel a report

Vyhodnocování klasifikačních modelů dobývání znalostí z databází

- Dostupná data se rozdělí na trénovací a testovací množinu
- Na trénovací množině se provede naučení klasifikátoru (rozhodovacího stromu)
- Na testovací množině se ověří jeho kvalita
- Validační data jsou již skutečná data

Kvalitu klasifikačního modelu lze ověřit pomocí metrik:

- Správnost (accuracy)
- Přesnost (precision)
- Úplnost (recall)
- Senzitivita
- Specificita
- Spolehlivost klasifikace
- Křivka učení
- Křivka navýšení
- Křivka ROC

Testování modelů

Při hledání znalostí pro potřeby klasifikace se obvykle postupuje metodou učení s učitelem. Vychází se tedy z toho, že jsou k dispozici příklady, o kterých víme, do které třídy patří. Metody evaluace jsou pak založeny na testování nalezených znalostí na datech, na možnosti porovnat, jak dobře se nalezené znalosti shodují s informací od učitele. Pro testování se nabízí celá řada variant podle toho, jaká data použijeme pro učení a jaká pro testování:

- testování v celých trénovacích datech
- křížová validace (cross-validation)
- leave-one-out
- bootstrap
- testování na testovacích datech

Testování v celých trénovacích datech

- testování na trénovacích datech (data použitá pro učení)
- nejmenší vypovídající schopnost o tom, jak budou nalezené znalosti použitelné pro klasifikování nových případů (kvůli overfitting)
- dochází často k přeučení (overfitting) – nalezené znalosti vystihují spíše náhodné charakteristiky trénovacích dat, než aby odhalili podstatné, co lze použít pro generalizaci

Křížová validace

- řeší problém - čím větší je část dat odložená pro validaci, tím lepší je odhad kvality modelu, ale nevýhodou je, že se zhoršuje kvalita stromu, protože odložená část dat nemůže být použita pro trénování
- data se rozdělí dopředu na několik částí (obvykle se rozděluje na 10 nebo 3 šásti)
- jedna část z celku, která slouží pro otestování chyby se vyjme (1/10) a zbytek (9/10) se použije pro učení, postup se opakuje podle počtu částí (10x) a výsledek testování se zprůměruje
- tomuto způsobu se říká n-násobná křížová validace (n-počet částí, desetinásobná)

Leave-one-out

- z dat, která jsou k dispozici se vyjme jeden příklad pro testování a zbylá data se použijí pro učení
- toto se opakuje tolikrát, kolik příkladů máme
- vznikne tedy n souborů znalostí, které se otestují na n příkladech
- výsledek testování dává odhad, jak by se znalosti získané ze všech dostupných n příkladů chovaly při klasifikování příkladů neznámých
- při velkém počtu příkladů je představa běhu algoritmu absurdní – existují ale algoritmy, co implementují leaveone out efektivněji např. ESOD

Bootstrap

- vybrané příklady pro učení se mohou opakovat
- oproti křížové validaci, kdy je příklad buď pro učení nebo pro testování, v bootstrapu lze tentýž příklad použít pro učení několikrát
- máme-li k dispozici n příkladů, provedeme n-krát výběr s navracením, abychom získali n příkladů pro učení
- pravděpodobnost, že bude příklad vybrán je 1/n a že nebude 1-1/n
- při n opakováních je pravděpodobnost vybrání e-1 = 0.368
- pro rozumně velká data se tedy vybere 63.2% příkladů pro učení a 36.8% pro testování (v praxi se většinou proto rozdělují data v poměru 70/30, 75/25, 80/20 apod.)

Ať už je využitá jakákoliv metoda, cílem testování je určit, v kolika případech se klasifikátor shoduje s učitelem a v kolika případech se dopustil chyb. Tyto údaje bývá zvykem zachycovat v tzv. matici záměn (confusion matrix)

STRUKTURA STROMU

Kořen, větve, uzly a listy. Počet větví vycházejících z uzlu atributu odpovídá počtu jeho kategorií (vícekategoriální atribut má mnoho větví, binární atribut má dvě větve – binární strom). Procházení do hloubky a do šířky (depth-first walking, breadth-first walking).

KVALITA MODELU

Na trénovacích datech

- čistota uzlu (purity) - počet objektů s převažující třídou ku počtu všech objektů patřících do daného uzlu
- kvalita stromu a podstromu -vážený součet čistot uzlů ve stromu nebo uzlů patřících do podstromu

Kvalita na testovacích datech

- vyzkoušení klasifikace pro další objekty, nepoužité při vytváření stromu

INDUKCE ROZHODOVACÍCH STROMŮ

Algoritmus “Top-down induction of decision trees” (TDIDT) - příklady: ID3, C4.5, CART

Vstupy

- množina atributů, které mohou být ve stromu jako větvící
- cílový atribut (třídy) nebo jedna cílová třída
- požadovaná čistota uzlu

Výstupy

- rozhodovací strom přiřazující záznamy analyzovaných dat do cílových tříd

Postup

Začneme jediným uzlem (kořenem). Z aktuálně dostupných atributů vybereme ten, v jehož jednotlivých kategoriích převažuje pro danou úroveň stromu co nejvíce vždy jedna cílová třída. Přidáme pod-uzly pro všechny kategorie vybraného atributu.

Nejznámější je metodika CRISP-DM = Cross-Industry Standard Process for Data Mining.

Fáze CRISP-DM:

1. **porozumění doménové oblasti**
    - seznámení s doménovou oblastí
    - zjištění cílů analýzy, jejichž splnění zadavatel očekává (také zhodnocení, jestli jsou jeho cíle reálné k získání)
2. **porozumění datům**
    - získání dat a jejich porozumění (struktura, rozsah, format)
    - Zhodnocení kvality dat (úplnost, chybně vyplněné informace, chybějící hodnoty)
3. **příprava dat**
    - jde o přípravu do podoby, která vyhovuje cílům analýzy, tak i nástrojům, které hodláme použít
    - součástí je transformace či ošetření chybějících a chybných hodnot
    - Transformace dat: diskretizace (všechny hodnoty rozdělíme do interválů), výpočet odvozených hodnot (např. výpočet BMI = body mass index z dat o výšce a váhe)
    - doplnění externích dat
4. **modelování (analytické procedury) - samotný data-mining**
    - aplikace analytických nástrojů
    - nástroje, které při analýze použijeme, by měly odpovídat cílům analýzy, povaze dat (jejich strukturovanosti atd.) a použité metodě
5. **vyhodnocení výsledků**
    - nutné vybrat jen ty výsledky pro majitele/zadavatele zajímavé
    - třeba upravit výstupy získané z nástroje, aby byly pro majitele/zadavatele pochopitelné (např. i pomocí různých způsobů vizualizace)
6. **využití výsledků**
    - klíčová fáze DZD
    - během předchozích fází/komunikace je třeba získat důvěru majitele/doménového experta, aby se výsledky, které jsme získali, řídil, případně k nim nějakým způsobem přihlédl

Typické úlohy DZD:

- **Explorační analýza (průzkum)** dat - zjištění všeho zajímavého v datech, výstupem je analytický model (popis dat) (opak: konfirmační analýza). Rozdíl mezi klasifikací a predikcí spočívá v tom, že u predikce hraje důležitou roli čas; ze starších hodnot nějaké veličiny se pokoušíme odhadnout její vývoj v budoucnosti
- **Klasifikace/Predikce** - posuzování nových objektů na základě poznání dřívějších/ na základě předem vytvořeného klasifikačního modelu.

Metody a techniky DZD:

Základní:

- Frekvenční a kontingenční analýza,
- Zhlukování,
- asociační pravidla,
- rozhodovací a explorační stromy.

Další: neuronové sítě, bayesovská klasifikace, rozhodování podle analogie (Case-based reasoning - CBR), evoluční algoritmy…

Pracnost jednotlivých částí modelování:

- diskuze s majitelem dat – cíle, smysl a cena analýzy, budování důvěry (25%)
- problém se získáním dat a jejich popisu (10%)
- čištění a předzpracování dat (25%)
- modelování (5%)
- co výsledky znamenají? (10%)
- důvěra ve výsledky? jak je použít? podstoupí riziko uvedení do praxe? (25%)

Za nejpracnější se ale obecně považuje příprava dat. K této fázi se často vracíme, abychom měli data pro analýzu správně a ideálně zpracována (rozdělena do skupin apod.) a zjednodušila se tedy analýza samotná. Jde tedy o zejména předzpracování, ale i čištění dat.

21. Ukládání a vyhledávání textových informací

Booleovský model vyhledávání dokumentů

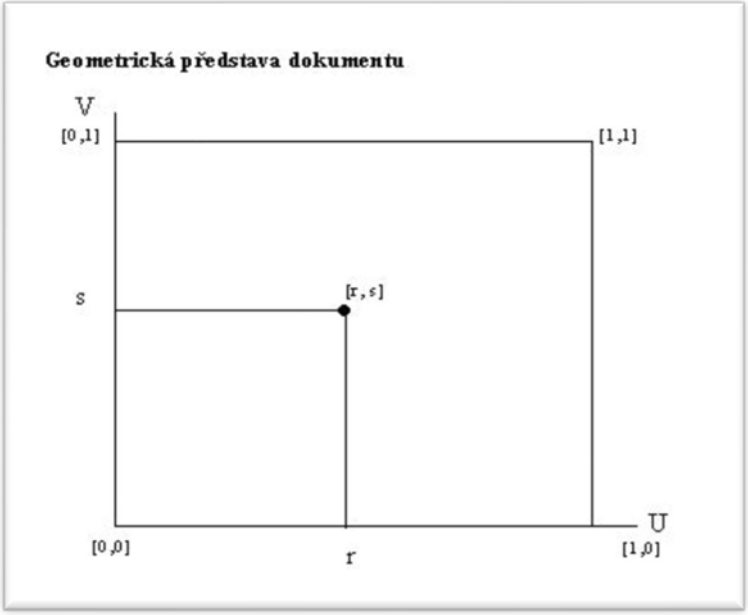
- vyhledáváme dokumenty pomocí klíčových slov, která můžeme specifikovat použitím logických spojek jako např. AND – vyhledá dokument, ve kterém se vyskytují obě klíčová slova, nebo OR – vyhledá dokumenty, ve kterých se vyskytuje jedno nebo obě tato klíčová slova, nebo NOT – které vyhledá dokumenty, ve kterých se vyskytuje první klíčové slovo, ale druhé ne
- Tento model vyhledává dokumenty podle pravdivosti a nepravdivosti výroků
- Příklady:
    - nedokonalé vyjadřovací schopnosti klíčových slov ((datová struktura OR algoritmus) AND CD-ROM)
    - indexátor nezná dokonale příslušnou oblast, a tak použije obecný termín, který je ale nedostačující, nebo uživatel použije jiné vyjádření daného slova než použil indexátor, řešením může být použití Tezauru (ten za nás automaticky doplňuje synonyma vyhledávaného slova)
- při tomhle způsobu vyhledávání se nedají odlišit jednotlivé stupně shody, buď jsou relevantní nebo ne
- nepřímá úměrnost mezi přesností a úplností vyhledávání

Rozšíření booleovs. modelu pomocí FUZZY logiky

- při booleovském vyhledávání se vyhledávají výrazy podle klíčových slov, a to na základě toho, jestli tam daný výraz je nebo není, číselně buď 1 nebo 0
- u fuzzy logiky můžeme těmto výrazům přidělit nějakou váhu, to znamená, že přiřazujeme, jakou pravdivost má daný výrok, tato pravdivost se značí Pr a může být z intervalu <0,1>
- např. Pr(„Míč je velký“)=0,7
- udává tak vlastně důležitost slov
- pravidla pro váhu klíčových slov v dokumentu:
    - Pr(A OR B) = max (Pr(A), Pr(B))
    - Pr(A AND B) = min(Pr(A), Pr(B))
    - Pr(NON A)= 1 – Pr(A)
- hodnotu Pr můžeme považovat za stupeň relevance určitého dokumentu k určitému dotazu, na základě toho pak můžeme rozhodovat o relevanci jednotlivých dokumentů
- celkový význam slova v dotazu i dokumentu se dá spočítat jako součin obou vah
- problémem je, že na výslednou hodnotu mají vliv nejvíce extrémní hodnoty, což se dá odstranit pomocí geometrického rozšíření
- řeší důležitost deskriptorů (slov) v dokumentu i dotazu a umožňuje řazení podle důležitosti
- neřeší tvrdost booleovských operací (AND) – od toho je geometrické rozšíření

Geometrické rozšíření booleovs.modelu

- řeší odstranění tvrdosti booleovských operace
- při booleovském vyhledávání se vyhledávají výrazy podle klíčových slov a to na základě toho, jestli tam daný výraz je nebo není, číselně buď 1 nebo 0
- stejně jako u fuzzy logiky můžeme těmto výrazům přidělit nějakou váhu, narozdíl od ní ale geometrické používá různé předpisy pro výpočet těchto vah
- u geometrického rozšíření si představujeme, že váha dokumentu je nějaký bod v prostoru a počet rozměrů tohoto prostoru závisí na počtu zvolených klíčových slov, a my zjišťujeme, jak daleko tento dokument leží ve čtverci, který má souřadnice [0,0],[0,1],[1,0] a [1,1]
- souřadnice r je váha klíčového slova U v dokumentu, souřadnice s je váha klíčového slova V v dokumentu
- v případě U OR V je pro nás lepší, čím dál je dokument o souřadnicích [r,s] vzdálen od bodu [0,0]
- v případě U AND V je pro nás lepší, čím blíž je dokument od bodu [1,1]
- při porovnávání booleovského vyhledávání, fuzzy a geometrického se došlo k výsledkům, že když se použije fuzzy, tak to ještě nutně nemusí znamenat zlepšení, naopak u geometrického rozšíření docházelo ke zlepšení vždy, všechny tyto metody jsou ale pořád nedokonalé



Přesnost vyhledávání = poměr relevantních vyhledaných dokumentů ku všem vyhledaným dokumentům

Úplnost vyhledávání = poměr relevantních vyhledaných dokumentů ku všem relevantním dokumentům

Vektorový model vyhledávání textových dokumentů

- stejně jako u fuzzy nebo geometrického rozšíření chceme vyhledávat dokumenty podle klíčových slov, kdy každému klíčovému slovu, jak v dokumentu, tak v dotazu, přiřadíme nějakou váhu
- v tomto případě si představujeme dokument i dotaz jako vektory, a jednotlivé prvky těchto vektorů jsou váhy klíčových slov
- v tomto případě se nedají vztahy klíčových slov vyjádřit OR, AND a NOT, ale zase se v průběhu vyhledávání dají měnit požadavky podle dokumentů, které už vyhledány byly, protože pokud jsou dokumenty relevantní, pak by také měly mít vzájemně podobné vektory
- dokumenty jsou vyhledávány na základě podobnosti s dotazem, a nakonec se dokumenty seřadí podle stupně podobnosti používají se různé vzorečky na výpočet této podobnosti např. Kosinová míra podobnosti, nebo Diceova míra podobnosti
- nevýhodou je, že chybí teoretické zdůvodnění výběru vhodné míry podobnosti

Dokument lze chápat i jako vektor. Předpokladem k tomu je, že obsah dokumentu je popsán klíčovými slovy a každému klíčovému slovu je přiřazena váha vyjadřující důležitost slova pro charakteristiku obsahu dokumentu. Předpokládáme i váhy klíčových slov v dotazu. Základní ideou vektorového přístupu je vyjádřit každý dokument i dotaz jako vektory, jejichž složky jsou váhy jednotlivých klíčových slov, a využít prostředků vektorového počtu k výpočtu podobnosti dotazu a dokumentu.

V případě geometrického rozšíření chápeme dokument D jako bod PD v prostoru. Souřadnice bodu PD jsou dány vahami jednotlivých klíčových slov u dokumentu D. To je prakticky stejný přístup jako u vektorového modelu, bod PD jednoznačně odpovídá vektoru s počátečním bodem o souřadnicích <0, 0,...,0> a koncovým bodem PD. Rozdíl je ve výpočtu podobnosti dotazu a dokumentu. Prostředky, které vektorový počet používá pro výpočet podobnosti vektorů, neumožňují vyjádřit vztahy dané logickými spojkami AND, OR a NOT, které jsou k dispozici v geometrickém rozšíření booleovského modelu. Na druhé straně je při vektorovém přístupu možno snadno průběžně modifikovat dotaz na základě již vyhledaných dokumentů.

Předpokládejme, že máme informační fond s N dokumenty D1 , D2 , ...DN , k jejichž indexování byla použita klíčová slova S1 , S2 , ..., SK. Váhu slova Sj v dokumentu Di značíme wi,j. K dokumentu Di je tedy přiřazen vektor <wi, 1, wi,2 , ..., wi, K>.

Zde je třeba upozornit, že z důvodů formálního vyjádření se uvádí váha pro každé ze slov S1 , S2 ,..., SK, včetně nulových vah. Pokud váha některého slova pro dokument je nulová, znamená to, že toto slovo nepatří mezi klíčová slova dokumentu, a ani trochu necharakterizuje jeho obsah. Například, je-li W2,1 = 0, znamená to, že slovo S1 není klíčovým slovem pro dokument D2. Fakt, že slovo S1 není klíčovým slovem pro dokument D2, je informace, která je využívána při výpočtu podobnosti dokumentu a dotazu. Pokud bychom u dokumentu uváděli jenom jeho klíčová slova s nenulovými vahami, narazili bychom na formální potíže při vyjadřování vektorových operací.

Jak už bylo uvedeno, jako vektor vah pro klíčová slova vyjádříme nejen každý dokument, ale i dotaz. Formálně budeme pro dotaz Q psát Q =<q1, q2, ..., qK>. Čísla q1, q2, ..., qK patří do intervalu <0, 1>. Uvedený zápis znamená, že váha klíčového slova S1 v dotazu Q je q1, váha klíčového slova S2 je q2 , atd.

Při vyhledávání pomocí vektorového modelu se pro každý dokument spočítá jeho podobnost s dotazem. Výsledkem vyhledávání je seznam dokumentů seřazený podle stupně podobnosti. Pro výpočet se používají obecné míry podobnosti vektorů. Takových měr existuje celá řada. Uvedeme dva příklady takových měr, Kosinovou míru podobnosti a Diceovu míru podobnosti

Nasledujúci vzorce platí pro dokument a dotaz **D = <w1, w2, ..., wK>** **Q = <q1, q2, ..., qK>**

$$\frac{\sum_{i=1}^{K} w_i * q_i}{\sqrt{\sum_{i=1}^{K} w_i^2 + \sum_{i=1}^{K} q_i^2}}$$

Diceova míra podobnosti se počítá jako:

$$\frac{2 * \sum_{i=1}^{K} w_i * q_i}{\sum_{i=1}^{K} w_i^2 + \sum_{i=1}^{K} q_i^2}$$

Příklad, Kosinova míra:

	Deskriptory			podobnost
	Tezaurus	Počítač	CD-ROM	dokument - dotaz
Dotaz	1.0	0.8	0.9	
Dokument A	0.9	0.1	0.8	0.90
Dotaz	1.0	0.8	0.9	
Dokument B	0.0	0.7	0.7	0.77
Dotaz	1.0	0.8	0.9	
Dokument C	0.7	0.9	0.8	0.99
Dotaz	1.0	0.8	0.9	
Dokument D	0.1	1.0	1.0	0.81
Dotaz	1.0	0.8	0.9	
Dokument E	1.0	0.8	0.9	1.00

Výhodou vektorového přístupu je relativně snadná možnost iterativní modifikace dotazu na základě již vybraných relevantních dokumentů. Modifikace vychází z předpokladu, že jestliže jsou dokumenty relevantní k dotazu, pak si jsou jejich vektory vzájemně podobné. Jestliže v průběhu vyhledávání získáme několik relevantních dokumentů, je možno automaticky modifikovat vektor dotazu tak, aby se co nejvíce podobal vektorům relevantních dokumentů. Použití přeformulovaného dotazu dává šanci na získání dalších relevantních dokumentů.

Mezi nevýhody vektorového přístupu patří chybějící teoretické zdůvodnění výběru vhodné míry podobnosti. Nevýhody plynou i z toho, že jednotlivá klíčová slova nejsou nezávislá. Jestliže se například ve fondu technické literatury vyskytuje u dokumentu jako klíčové slovo termín ”bezpečnostní pasy” tak se zvyšuje šance, že se bude vyskytovat i klíčové slovo ”pasivní bezpečnost”. Z toho vyplývá, že příslušné složky vektoru dokumentu nebudou ortogonální (kolmé), což bude komplikovat vektorové operace.

22. Regulární výrazy

Regulární výraz (regular expression, dále jen regexp) slouží k vyhledání části řetězce, kterou předem (úplně) neznáme nebo která může mít více podob. Používá se v programovacích a skriptovacích jazycích.

Regulární výrazy vznikly z důvodu potřeby práce s textovými řetězci určitým unifikovaným způsobem. Jsou zajímavý nástroj nejen pro ověření, zdali zadaný textový řetězec splňuje určená pravidla (validace), ale také nám umožňují vyhledávat určité podřetězce poměrně jednoduchým způsobem. Zbavíme se tak mnohdy i několika vnořených podmínek.

Regulární výraz je textový řetězec složený z určitých znaků. Neznám nikoho, kdo by si tento řetězec přečetl a ihned pochopil, co daný výraz znamená. Gramatika regulárních výrazů není složitá, ale je poměrně nepřehledná a proto je dobré již napsané výrazy komentovat.

[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}

- Cílem tohoto regulárního výrazu je zjednodušeně zjistit, zda-li je vložený textový řetězec emailem. Výraz je dost zjednodušený, takže některé neplatné adresy jím projdou.

Unix bez regulárních výrazů je jako sex bez partnera/partnerky. Dá se to požívat, ale člověk o cosi zásadního přichází. Znalost regulárních výrazů vám da do rukou mimořádně silný nastroj pro práci s textem. Jejich prostřednictvím můžete:

- vytahovat z textových dat údaje, které vás zajímají
- přetvářet je do podoby, kterou potřebujete
- vyhledávat a nahrazovat v textových editorech a dalších programech

Existují různé odnože regulárních výrazů – nejznámější dvě odnože jsou Perl-compatible regulání výrazy a **POSIX regulární výrazy.** Ačkoliv většina novějších programovacích jazyků používá regulární výrazy odvozené od jazyka Perl, drobné rozdíly v implementaci tu stále přetrvávají (například Javascript nepodporuje všechny Perl regular expressions konstrukce).

Kvantifikátory

Kvantifikátor	Počet opakování
?	minimálně 0krát, maximálně 1krát
*****	minimálně 0krát (maximálně neomezeno)
+	minimálně 1krát (maximálně neomezeno)
{n}	právě *n*krát
{m,n}	minimálně *m*krát, maximálně *n*krát
{m, }	minimálně *m*krát (maximálně neomezeno)

- Existují tzv**. liné kvantifikátory** – ty se od výše zmíněných (tzv. nenasytných) liší v zápisu tak, že výše uvedený kvantifikátor zprava doplníme o otazník (?). Líné kvantifikátory tedy budou ??, *?, +?, {m,n}? a {m,}?.
- Funkčně se budou líné kvantifikátory (od v tabulce uvedených nenasytných kvantifikátorů) lišit v tom, že pomocí líných kvantifikátorů je zachycen minimální počet znaků, které je třeba zachytit, aby došlo ke shodě s regulárním výrazem. Nenasytné kvantifikátory naopak zachytí co možná největší počet znaků vstupního textu.

Předdefinované skupiny znaků

Znak	Popis
\d	Číslice 0-9
\D	Jakýkoliv znak, kromě číslic 0-9
\w	Znaky „slova" (ekvivalentní zápisu [a-zA- Z0-9_])
W	Jakýkoliv znak kromě znaků „slova" (ekvivalentní zápisu [^a- zA-Z0-9_])
\s	„bílé" znaky (mezera, tabulátor, znaky pro zalomení řádků)
\S	Jakýkoliv znak kromě „bílých" znaků

Hranice

Znak	Popis
^	začátek řetězce (textu v němž se vyhledává)
$	Konec řetězce (textu v němž se vyhledává)

Původní význam znaků (metaznaků)

Možná vás napadlo, že když určité znaky (metaznaky) mají v regulárním výrazu zvláštní význam, jak je možné takový znak zapsat tak, aby nebyl chápán jako metaznak, ale jako obyčejný znak (třeba plus či hvězdička). Řešení je prosté –

stačí před inkriminovaný znak dopnit v regulárním výrazu zpětné lomítko \. Pokud chcete například pomocí regulárního výrazu popsat rovnici (a+b)*c=d, je třeba použít regulární výraz (a\+b)*c=d.

Které znaky je třeba doplnit oním zpětným lomítkem (tzv. escapovat)? Mezi metaznaky patří \, ^, $, ., [,], |, (,), ?, *, +, {, }.

Formální definice

Formálně je regulární výraz definován následujícím způsobem:

1. a je regulární výraz pro libovolný znakový literál (znak abecedy) a, popisující právě text a.
2. ε je regulární výraz pro prázdný řetězec.
3. Znak pro prázdnou množinu označuje prázdný jazyk.
4. Pokud A a B jsou regulární výrazy, je AB regulární výraz, popisující zřetězení textů popsaných výrazy A a B.
5. Pokud A a B jsou regulární výrazy, je A + B regulární výraz, popisující buď text popsaný výrazem A, nebo text, popsaný výrazem B.
6. Pokud A je regulární výraz, pak A* je regulární výraz, popisující libovolný počet opakování (včetně žádného opakování) textů popsaných výrazem A.
7. Pokud A je regulární výraz, je (A) regulární výraz popisující stejný jazyk. (Závorky slouží pouze pro vyjasnění priorit.)

Příklad

Regulární výraz	Odpovídá...
`a+`	sekvence písmen `a` (1 a více znaků)
`a*`	sekvence písmen `a` (0 a více znaků)
`o?kov`	`okov` či `kov`
`tel(efon)?`	`tel` či `telefon`
`telef(on|ax)`	`telefon` či `telefax`
`[0-9]|[1-9][0-9]`	čísla 0 až 99
`\d{2}`	sekvence dvou číslic desítkové soustavy (`00`, `01`, ..., `98`, `99`)
`[0-9a-fA-F]|[1-9a-fA-F][0-9a-fA-F]+`	hexadecimální čísla
`(19|20)\d{2}`	letopočty 1900-2099
`\d{2,6}`	sekvence dvou až šesti číslic
`[^ ,.]+`	neprázdná sekvence znaků mezi nimiž nesmí být mezera ( ), čárka (`,`) či tečka (`.`)
`^P.*`	řetězec, který začíná písmenem `P` za nímž následuje libovolný (i nulový) počet libovolných znaků
`\d+0$`	řetězec, který končí znakem `0` (nula), kterému předchází minimálně jedna číslice
`a+b`	`ab`, `aab`, `aaab` atd.
`a\+b`	`a+b`

23. Rozdíl mezi Governance a Management

Rozdíl mezi Governance a Management je zásadním pojmem v oblasti řízení organizace a jejích procesů. Tyto dva koncepty se liší nejen ve svých úkolech a funkcích, ale i v tom, jakým způsobem přispívají k celkovému řízení a směřování organizace.

Governance (řízení) zahrnuje celkovou strategii a směřování organizace. Jeho cílem je vytvářet a prosazovat pravidla, politiky, a procedury, které definují jak organizace funguje a jak dosahuje svých cílů. To zahrnuje stanovení etických standardů, právních předpisů a očekávání stakeholderů. Governance se stará o to, aby organizace jednala v souladu s těmito principy, a zajistí transparentnost a odpovědnost. Jeho úrovně rozhodování jsou strategické a vedou k určení, co organizace má dělat a proč.

Na druhou stranu **Management** se zaměřuje na provádění strategií a plánů definovaných v rámci governance. Jeho úkolem je řídit každodenní operace organizace a zajistit, že stanovené cíle jsou dosaženy. To zahrnuje plánování, organizaci, řízení lidí, procesů a technologií. Management se zabývá taktickými a operativními rozhodnutími, která vedou k dosažení cílů stanovených v rámci governance. Zatímco governance určuje směr organizace, management implementuje konkrétní akce, které vedou k dosažení těchto cílů.

V praxi, Governance a Management **spolupracují a doplňují se navzájem**. Governance poskytuje rámec a směrování, zatímco management provádí strategie a plány definované v tomto rámci. Jejich úspěšná spolupráce je klíčová pro dosažení cílů organizace a zajištění efektivity a odpovědnosti ve všech oblastech činnosti.

24. Klíčové oblasti Enterprise Governance of IT

Enterprise governance

=správa podniků

ENTERPRISE GOVERNANCE – množina odpovědností a postupů prováděných představenstvem a exekutivou s cílem realizovat: strategické cíle, omezit rizika a zajistit, že podnikové zdroje se využívají efektivně.

je možné definovat jako „seznam zodpovědností a praktik vykonávané představenstvem společnosti a výkonným managementem za účelem:

- **poskytování strategických rozhodnutí**
- **dosažení hlavních cílů společnosti**
- **zajištění správného řízení rizik a ujištění, že zdroje společnosti jsou využívány zodpovědně**

Enterprise governance tvoří základní rámec zodpovědnosti pro každou společnost. Tento rámec je možné rozložit na dvě základní roviny, které by měly být v rovnováze, a to **soulad a výkonnost**.

Oblast soulad pokrývá:

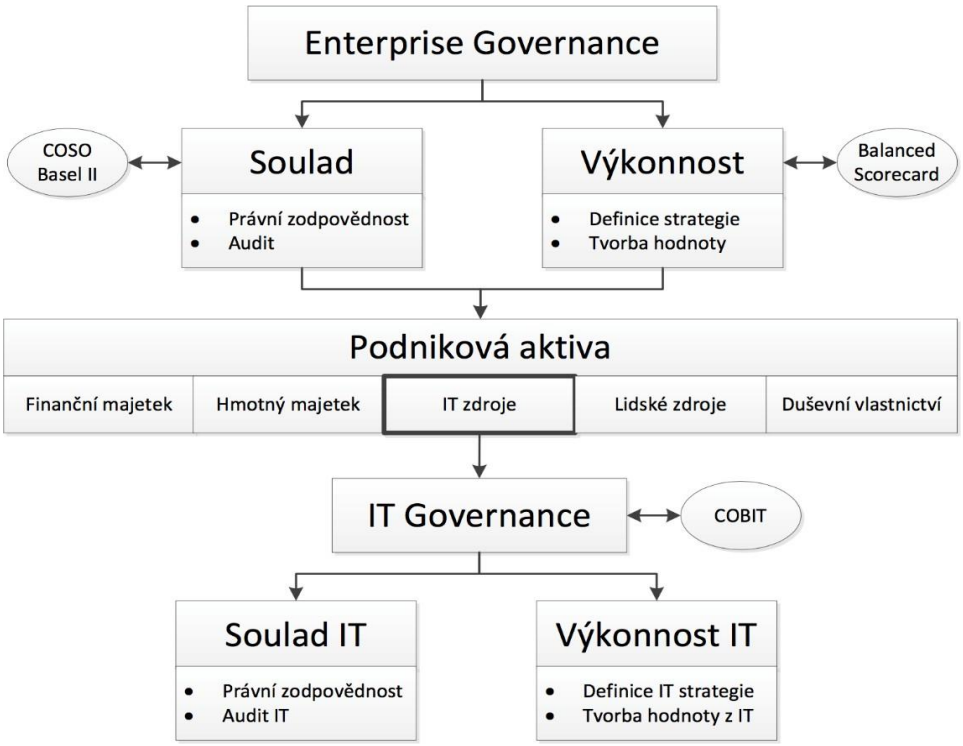
- Zodpovědnost a role výkonného ředitele (CEO)
- Role a zodpovědnost představenstva nebo jiného statutárního orgánu
- Role a zodpovědnost dozorčí rady, valné hromady atd.
- Požadavky na interní kontroly ve společnosti
- Řízení rizik a interní audit a
- Systém odměn a platů

Oblast výkonost pokrývá

- Úspěšnost společnosti spočívá v prvé řadě na správné definici strategie a provedení akcí, které směrují k naplnění strategických cílů. **Provádění strategických rozhodnutí**.
- Dále je nutno zajistit tvorbu hodnot, které očekávají zůčastněné strany a to především akcionáři a zákaznící a to při vhodném užití podnikových zdrojů. **Určení hlavních výkonnostních ukazatelů**.

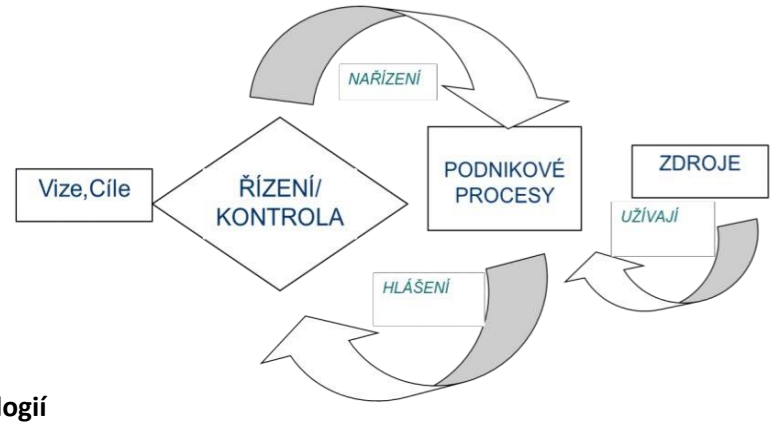
- Využívání zdrojů a provádění strategie sebou nese určitá rizika. Tyto rizika je nutné vhodným zúůsobem zmírnit, odstranit či akceptovat v závislosti na ochotě podniku nést vyšší nebo nižší rizika spojené s podnikáním. **Pochopení určení přístupu k rizikům**.

Rozměr výkonosti je mnohem komplexnější než dosažení souladu se standardy a provádění auditů.



Governance proces

1. akcionáři, statutární orgány společně se zástupci exekutivy stanoví podnikové cíle, vize,strategie,
2. vedení podniků se potom tyto cíle za pomoci obecně uznávaných (a pokud možno nejlepších) zkušeností a postupů řízení snaží naplnit,
3. na základě stanovených cílů se formulují nařízení (pokyny), která ovlivňují podnikové aktivity a spotřebovávají zdroje,
4. výsledky podnikových procesů se měří a hlásí prostřednictvím systému kontrol.



IT Governance

= správa informačních technologií

IT Governance je definovaná struktura vztahů a procesů, pomocí kterých lze řídit a kontrolovat organizaci tak, aby IT v maximální míře umožňovalo a podporovalo dosažení podnikatelských cílů. IT Governance ovlivňuje zvýšení efektivity organizace pomocí:

- Zajištění a zabezpečení integrity, bezpečnosti a spolehlivosti strategických a jiných citlivých informací.
- Ochrany investic do IT a komunikací
- Nastavení odpovídajícího vedení a řízení informačních aktiv, na nichž přímo závisí úspěch nebo přežití organizace
- Zvyšování hodnoty podnikatelských procesů pomocí IT (vazba IT na podnikatelské procesy)

IT Governance je označení přístupu a způsobu řízení IT procesů v organizaci, který slaďuje informační systém a všechny informační technologie s globální strategií organizace.

IT Governance v dlouhodobém horizontu zajišťuje správné cílení všech investic do informačního systému organizace díky jasně definovaným procesům posuzování všech strategických požadavků na rozvoj.

IT Governance je ve skutečnosti soubor pravidel, vztahů a procesů, které pomáhají řídit organizaci tak, aby IT v maximální míře podporovalo její strategické cíle.

- **Důvody zavedení IT governance:** IT je významným kritickým faktorem úspěchu pro dosažení podnikové strategie, IT je prvek, který umožňuje růst a vývoj podniku, IT musí splňovat stále rostoucí nároky v oblasti regulací, povinné správy a ochrany IT aktiv
- **Přínosy nasazení IT governance:** Podnik dostává vyšší stupeň důvěry IT, Zdokumentované IT procesy a role

Hlavní rámce pro IT Governance

IT governance zajišťuje vykazovací povinnost a poskytuje platformu k implementaci best practice podle standardů, jakými jsou například **ITIL** (Information Technology Infrastructure Library) či **COBIT**. Kontrolní rámce (například Cobit), jež pomohou IT organizacím pochopit, jak měřit a kontrolovat IT procesy**. Best Practices** (například ITIL) se snaží IT organizacím pomoci implementovat procesy a kontroly identifikované v předchozích dvou bodech.

Oblasti IT governance

- **Sladění a synchronizace strategií v rámci podniku**
  Hlavní otázkou je, zda jsou investice do IT v souladu s podnikovou strategií a cíly. Proces harmonizace podnikové a IT strategie (byznys a IT procesů) je nikdy nekončící proces – důležitá je tedy především permanentní snaha o přiblížení a vyrovnání uvedených prvků.
- **Generování hodnoty**
  Představuje realizaci přidané hodnoty pomocí IT, tj. realizaci slíbených přínosů k dané strategii s důrazem na optimalizaci nákladů a vytváření hodnot charakteristických pro IT.
- **Řízení zdrojů**
  Znamená optimalizaci IT investic a správné řízení kritických IT zdrojů: aplikací, informací, infrastruktury a lidí. Klíčové jsou pro tuto oblast optimalizace znalostí a infrastruktura.
- **Risk managament – řízení rizik**

25. Základní principy COBIT 5

COBIT (Control Objectives for Information and Related Technologies)

- Framework vytvořený mezinárodní asociací ISACA pro správu a řízení informatiky (IT Governance). Jedná se o soubor praktik, které by měly umožnit dosažení strategických cílů organizace díky efektivnímu využití dostupných zdrojů a minimalizaci IT rizik. Prakticky je tedy určen především top manažerům k posuzování fungování ICT a auditorovi pro provádění auditu systému řízení ICT. Na rozdíl od ITIL, který je více určen více manažerovi IT (CIO).

Návody, jak zacházet s podnikovou architekturou, jak řídit informační aktiva a služby, jak zajistit správnou dodávku služeb v návaznosti na organizační modely a jak zacházet s vznikajícími trendy a technologiemi.

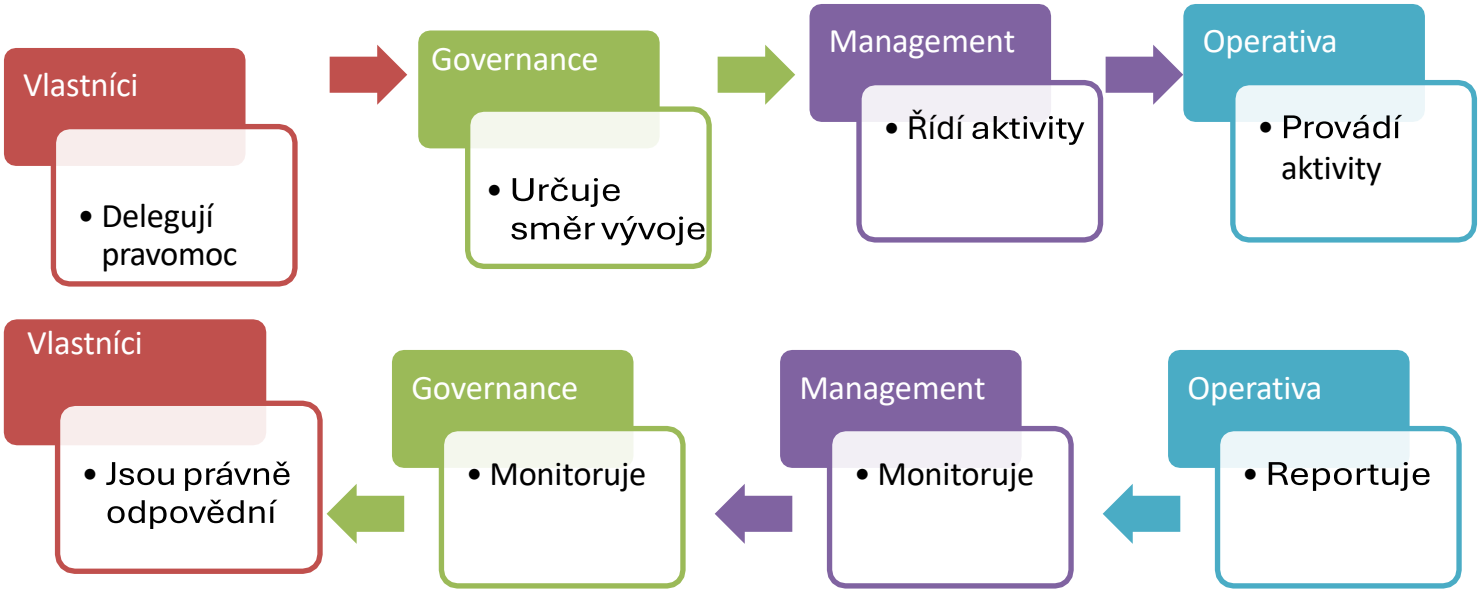
COBIT 5 - je založený na 5 principech:

- **Naplnění potřeb různých zainteresovaných osob**
  - Uspokojování potřeb zainteresovaných stran IT existuje pouze pro potřeby celé organizace. Bez této základní zásady nemá smysl správu informací realizovat.
- **Kompletní pokrytí celého podniku**

- Pokrytí celého podniku COBIT pokrývá využívání informací a IT v celém podniku, nikoliv pouze funkci IT. Správa informací je důležitá pro všechny.

– celý produkční, informační a řídící proces

- **Užití jednoho integrovaného rámce**
  - Použití jediného integrovaného rámce COBIT 5 je na vysoké úrovni v souladu s řadou dalších rámců a metodik, jako je ITIL® nebo ISO27001, a díky tomu může fungovat jako zastřešující rámec pro správu a řízení všech informací.
- **Umožnění holistického přístupu – hledat systémové souvztažnosti v celém podniku**
  - Umožnění holistického přístupu COBIT se domnívá, že souběžný pohled na širokou škálu dotčených oblastí je lepším přístupem než práce v "silách". Pohled na správu informací v široké škále zdrojů nebo "faktorů" je proto užitečnější než pohled pouze na jednu oblast.
- **Odlišení governance a management**
- Oddělení správy od řízení Správa a řízení nejsou totéž. Řízení je pochopení potřeb organizace, definování směru prostřednictvím stanovení priorit a rozhodování a sledování souladu s cíli. Řízení je mechanismus, jehož prostřednictvím jsou vytvářeny plány a probíhá v souladu s dohodnutými cíli. Řízení říká, co je třeba udělat, management se zaměřuje na to, jak to bude provedeno.



Ke každé metrice doplňuje KGI (Key Goal Indicator) a KPI (Key Perfomance Indicator).

Tato metodika rozděluje IS/ICT na jednotlivé funkční domény (plánování, implementace, provoz, monitoring) a v nich jednotlivé procesy poměřuje sedmi kritérii (efektivnost, výkonnost, důvěryhodnost, integrita, dostupnost, přizpůsobivost, spolehlivost). Výsledná zjištění přiřazuje 5 zdrojům (personál, aplikace, technologie, vybavenost, data). Výsledkem je normovaný pohled na způsob řízení podnikové informatiky a dosaženou úroveň služeb IS/ICT.

(Enterprise Governance – pravidla pro řízení celého podniku; CobiT je jen pro IT Governance)

Rozděluje IT do 4 domén, z nichž každá definuje nějaké IT procesy (celkem 34). Pro každý proces jsou navržena kritéria měření výkonnosti a kritéria hodnocení rizik.

- **Plánování a organizace**
- **Akvizice a implementace**
- **Poskytování a podpora**
- **Monitorování a vyhodnocení**

Lze ho využít pro sladění podnikové vize a IT cílů, také jako nástroj auditu podnikové informatiky. Při zavádění procesů lze použít již existující procesy nebo jiné (třeba podle ITIL). Při kontrole shody se SOX se vychází z rámce CobiT (SarbanesOxley Act – nutný pro veřejně obchodované společnosti v USA).

Využívá pro posuzování vyspělosti řízení informatického procesu tzv. Capability Maturity Model (CMM)

Zralost procesů = Capability Maturity Model (CMM)

Strukturovaný soubor praktik, jejichž aplikace vede ke zvyšování efektivity procesu. CMM rozlišuje 6 stupňů zralosti procesu:

- **Neexistující** – neexistuje žádný pozorovatelný proces. Organizace dosud nezpozorovala, že má problémy, které je potřeba řešit. Při výskytu události reaguje spontánně.
- **Náhodný** – organizace zjišťuje, že má problémy a pociťuje potřebu je řešit. Neexistuje konsolidovaný přístup, veškeré relevantní aktivity se provádějí na ad-hoc a individuální bázi.
- **Opakovaný** – ale pouze intuitivní – existuje snaha o vytvoření standardních procesů, jejich využití je však intuitivní, což vede například k tomu, že stejně činnosti jsou opakovány různými lidmi.
- **Formalizov**aný – proces je popsaný a standardizovaný, zaměstnanci jsou vyškolení v tom, jak proces vypadá a jaká je jejich role v procesu.
- **Měřitelný** – je přidán proces řízení a kontroly průběhu procesu. Na základě analýzy hodnot metrik se proces neustále zlepšuje.
- **Optimalizovaný** – proces byl vyvinut do jeho nejlepšího možného stavu na základě průběžného zlepšování a sledování „best practices" z okolí podniku. Činnosti zaměřené na optimalizaci procesu jsou součástí procesu.

SW-CMM – Model zralosti vývoje software (Capability Maturity Model for Software).

Organizace, které zaváděly více modelů hodnocení zralosti procesů, se dostávaly do problémů s jejich integrací. Proto byl vytvořen Capability Maturity Model Integration (CMMI), který spojuje a rozšiřuje výše uvedené modely.

26. Na co reaguje ITIL 4 a jaké jsou rozdíly vůči ITIL 3

ITIL (IT Infrastructure Library) - sada publikací popisujících nejlepší praktiky řízení ICT služeb + systém certifikací a školení jednotlivců ve znalostech těchto praktik. Doporučení nejsou závazná, každý čtenář může těchto nejlepších praktik využít podle libosti a svých potřeb.

ITIL® vznikl v 80. letech minulého století jako sada knižních publikací popisujících způsob řízení IT služeb a ICT infrastruktury. Přínos: jasná definice hlavních procesů správy IT služeb, rozvoj popisných pravidel. Řešení „na míru" – vycházíme z ITIL, ale přizpůsobujeme našim potřebám. Cíl zavedení –zlepšení kvality služeb, optimalizace, v zájmu firmy a jejich cílů (například obchodních).

ITIL je určen především IT manažerům. (ICT Management: taktické a operativní řízení, zaměřeno na každodenní schopnosti ICT plnit definované služby a plnit stanovené úkoly, především interní řízení útvarů ICT; ICT Governance: strategické a dlouhodobé řízení, zaměřeno na soulad ICT se strategickými zájmy organizace, vzájemné pochopení a oboustranná komunikace mezi ICT a ostatními útvary).

ITIL – sada publikací popisujících nejlepší praktiky řízení ICT služeb. Zároveň sada certifikací a školení.

Poskytuje rámec pro zvládnutí IT v organizaci, komplexně pojednává o službách zaměřuje se na neustálé měření a zlepšování kvality dodávaných služeb IT, a to jak z pohledu businessu, tak z pohledu zákazníka.

Aktuální je V3, která obsahuje pět ústředních publikací:

- ITIL Service Strategy (Strategie služeb)
- ITIL Service Design (Návrh služeb)
- ITIL Service Transition (Přechod služeb)
- ITIL Service Operation (Provoz služeb)
- ITIL Continual Service Improvement (Neustálé zlepšování služeb)

+ kniha The Introduction to the ITIL® Service Lifecycle, která ve stručnosti shrnuje principy uvedené v těchto pěti knihách.

Těchto pět publikací současně popisuje jednotlivé fáze **životního cyklu služby**. Co je životní cyklus služby? - Podle ITILu vychází podoba každé služby ze strategie (Service strategy), jež definuje důvody pro její existenci, následně služba prochází tvorbou návrhu (Service design), jenž je následně zrealizován, služba je nasazena do provozu (Service transition) a na každodenní bázi provozována (Service operation). Ve všech fázích životního cyklu služby pak dochází k neustálému zlepšování (Continual service improvement) všech aspektů služby.

ITIL v3 se zaměřuje na Service Lifecycle, který je rozdělen do pěti fází:

- **Service Strategy (Strategie služeb):** Zaměřuje se na plánování strategie služeb a určení, jaké služby by měla organizace nabízet a jakým způsobem by měly přispět k obchodním cílům.
- **Service Design (Návrh služeb):** Tato fáze se zabývá návrhem služeb, včetně jejich specifikace, architektury, procesů a technologií, aby byly schopny splnit požadavky zákazníků a podporovat strategii služeb.
- **Service Transition (Přechod služeb):** V této fázi jsou nové nebo upravené služby přecházeny do provozu. Zahrnuje testování, nasazení a přechodní období, aby se zajistilo, že změny budou úspěšně zavedeny a budou plně funkční.
- **Service Operation (Provoz služeb):** Service Operation se zabývá denním provozem služeb, včetně správy incidentů, problémů, změn a vydání. Cílem je zajistit, aby služby byly dostupné, spolehlivé a efektivní.
- **Continual Service Improvement (Trvalé zlepšování služeb):** Tato fáze se zaměřuje na průběžné zlepšování služeb a procesů na základě zpětné vazby a výkonnosti. Cílem je dosahovat stále lepších výsledků a poskytovat vyšší hodnotu zákazníkům.

ITIL 4 reaguje na měnící se podmínky v oblasti IT a businessu a přináší několik klíčových rozdílů oproti ITIL v3:

- **Flexibilita a adaptabilita:** ITIL 4 klade důraz na schopnost adaptace na změny v obchodním prostředí a technologických trendech. Začleněním Agile, DevOps a Lean konceptů má být ITSM framework lépe připraven k evoluci spolu s průmyslem.
- **Zaměření na hodnotu:** ITIL 4 zdůrazňuje vytváření hodnoty pro zákazníky a organizaci jako celek. Namísto pouhého řešení technických problémů se ITSM soustředí na dosahování obchodních cílů a poskytování služeb, které přinášejí skutečný užitek.
- **Inkorporace nových trendů:** ITIL 4 reflektuje nové trendy v oblasti IT, jako je dominace softwaru, Agile, Lean a DevOps přístupy. Tím se snaží být lépe přizpůsoben současným výzvám a potřebám organizací.

Celkově lze říci, že ITIL 4 přináší modernizovaný přístup k IT Service Managementu, který se snaží lépe reflektovat současnou dynamiku oboru a poskytovat flexibilnější a adaptabilnější rámec pro řízení služeb.

27. Klíčové aplikace – ERP, CRM, BI, SCM, MES

Jedná se o moduly rozšiřující funkcionalitu ERP na ERP II (Entrprice resource planning)

Tyto moduly jsou:

- ERP (Enterprise Resource Planning) – Podnikový IS
- SCM (Suply Chain Management) – Řízení dodavatelského řetězce
- CRM (Customer Relationship Management) – Řízení vztahu se zákazníkem
- BI (Business Intelligence) – Manažerský IS

Dodatečná specifická funkcionalita modulů

- PDM (Product Data Management) – Správa dat vztahující se k výrobku
- PLM (Product Lifecycle Management) – Řízení průběhu celého životního cyklu výrobku

- SRM (Suplier Relationship Management) – Řízení vztahu s dodavateli
- ERM (Employee Relationship Management) – Řízení vztahu se zaměstnanci

ERP (Enterprise Resource Planning)

- ERP představuje balíkový podnikový systém, který umožňuje automatizovat a integrovat většinu vnitropodnikových procesů a sdílet společná data v rámci celého podniku.
- Do ERP patří kompletní data o výrobku, logistika, finance a lidské zdroje.
- V ERP vzniká nejvíce dat o podniku, která potom ostatní moduly ERP II využívají.
- Je primárně určeno pro pořizování a aktualizaci dat.
- ERP je jádrem aplikační architektury podniku, pokrývá největší rozsah funkcí a procesů podniku.

Vlastnosti logistické části ERP

Pro výrobní a distribuční podniky je důležitá **podpora nákupu a odbytu**. Logistické procesy se spojují do celku a **zjednodušují a urychlují činnosti logistiky**. Zlepšují tok informací a tím i **rozhodování při plánování**. Integrují se i **systémy pro správu, plánování a řízení údržby**. Dále je pak významná podpora projektového řízení Integrují systémy pro plánování, správu a řízení údržby Podpora projektového řízení (tendence k individualizace zakázek pro jednotlivé zákazníky-stvá se projektem)

Řízení nákupů a skladů

- **řízení dodavatelů** – evidence a analýzy dodavatelů, analýzy dodavatelských cen
- **řízení nákupu** – evidence požadavků na materiál jednotlivých výrobních a dalších středisek
- **akumulace požadavků** na nákup, blokace materiálu, zpracování poptávek, evidence nabídek,
- **zpracování objednávek**, evidence dodacích listů, vytvoření příjemky materiálu na sklad
- **řízení skladových zásob** – evidence zásob, příjem a výdej ze skladu, inventury, měsíční uzávěrky skladu, výkazy skladu, obrazové soupisky zásob, likvidace nepotřebných zásob.

Výrobní modul ERP

Plánování výroby, výrobních zakázek, sledování jejich stavu, sledování a vyhodnocování skladových zásob, řízení výroby na úrovni operativního a dílenského řízení.

Výrobní modul poskytuje nástroje jako:

- kusovníky – evidence a správa kusovníkových položek a jejich charakteristik, přiřazení kusovníku k výrobní zakázce
- konfigurátor výrobku – možnost konfigurace vybraných výrobků, stanovení cenových údajů podle proměnných v modelech jednotlivých výrobků, stanovení dodacích termínů
- správa výrobních zakázek – zobrazení a sledování prodejních objednávek, zakládání výrobních zakázek, řízení nákupu od subdodavatelů, plánování jednotlivých výrobních zakázek
- prognózy a plánování výroby – optimalizace plánování výroby, plánování s respektováním omezených kapacit a dostupnosti materiálů, stanovení nejdříve možného dodání
- operativní plánování a řízení výroby

Finanční část ERP musí poskytovat komplexní pohled na finanční data v celé organizaci a efektivní provádění finančních operací Poskytuje komplexní přehled o finančních operacích v podniku, hodnocení ekonomické výkonnosti podniku i jeho jednotlivých obchodních jednotek a průběžné zajištění shody informačního systému s platnou legislativou. Jedná se o vedení všech finančních operací a sleduje se především účetnictví, a to zejména hlavní kniha, saldokonta dodavatelů a odběratelů, správa investičního majetku a finanční konsolidaci.

Data jsou do systému zapisována ze zdrojů, kterými jsou účetní doklady. Výsledkem jsou veškeré účty, rozvahy, výkazy zisků a ztrát. Vše bývá spojeno s ostatními daty ERP, takže máme vše dostupné v reálném čase a synchronizováno. Jsou implementovány i všeobecně uznávané postupy jako GAAP (Generally Accepted Accounting Principles). Dodavatelé ERP často zaručují soulad s legislativou, a to i účetní, takže máme jistotu, že účtujeme dle zákona.

Rozsah funkcionalit

- Finanční účetnictví
    - Hlavní kniha, pohledávky, závazky, konsolidace (spojení více závazků v jediný), pokladna, elektronický bankovní styk…
- Nákladové účetnictví
    - Účetnictví nákladových středisek, účetnictví ziskových středisek, nákladové účetnictví zakázek a projektů, zúčtování výkonů, procesní řízení, podpora ABC (Acitivity Based Costing).
- Controlling
    - Řízení nákladů, výnosů, zdrojů a termínů. Podrobné analýzy plánu a skutečnosti. Jedná se o klíčový nástroj pro strategické plánování.
- Investiční majetek
    - Správa a účtování investičního majetku, plánování a sledování nedokončených investic a investičních akcí – hospodaření s investicemi provází celý životní cyklus investičního majetku.
- Hotovost
    - Řízení hotovosti, předpověď likvidity, předpovědi cash-flow, finanční plánování a rozpočty, řízení rizik, peněžní obchody, měnové transakce a transakce s cennými papíry.
- Výpočet a účtování mezd
- Normy
    - Výkaznictví dle různých účetních norem (např. IAS – mezinárodní účetní standardy, IFRS – mezinárodní standardy účetního výkaznictví, GAAP – Generally Accepted Accounting Principles).
- Účtování v cizích měnách a kurzové rozdíly

SCM (Suply Chain Management)

- Aplikace podporující vazby podniku a jeho okolí.
- Zajišťuje řízení dodavatelských řetězců.
- Představuje soubor nástrojů a procesů, které slouží k optimalizaci a zefektivnění všech prvků dodavatelského řetězce.
- Propojuje komunikaci dodavatele a odběratele, a to pomocí ICT.

Samostatnými iniciativami v řízení dodavatelského řetězce jsou systémy pro správu dodavatelského řetězce (SCM). SCM systémy umožňují organizacím řídit tok zboží a informací od dodavatelů až po konečného zákazníka. Tyto systémy integrují různé operace, jako jsou plánování, nákup, výroba a distribuce, s cílem optimalizovat výkonnost dodavatelského řetězce jako celku.

SCM aplikace mají několik klíčových funkcí a výhod:

- **Plánování dodavatelského řetězce:** SCM systémy umožňují organizacím plánovat poptávku, zásoby a výrobu v souladu s potřebami trhu a zákazníků. To zahrnuje predikci poptávky, optimalizaci skladových zásob a plánování výroby.
- **Správa dodavatelů:** SCM aplikace umožňují organizacím spravovat vztahy s dodavateli, sledovat dodávky a zabezpečit dodržování dohodnutých podmínek a termínů dodávek. To pomáhá minimalizovat rizika spojená s nedostatečnou dostupností surovin nebo komponent.
- **Optimalizace logistiky:** SCM systémy pomáhají organizacím optimalizovat logistické procesy, včetně plánování tras, sledování dodání a řízení skladových zásob. To vede ke zlepšení efektivity a snížení nákladů spojených s distribucí zboží.
- **Sledování a analýza výkonu:** SCM aplikace poskytují organizacím možnost sledovat a analyzovat výkon dodavatelského řetězce, včetně časových prodlev, kvality dodávek a nákladů spojených s logistikou. To umožňuje identifikovat oblasti pro zlepšení a optimalizaci procesů.

Celkově lze říci, že SCM systémy jsou klíčovým nástrojem pro optimalizaci a řízení dodavatelského řetězce organizace. Pomáhají zlepšit efektivitu, snížit náklady a maximalizovat hodnotu pro zákazníky prostřednictvím lépe řízeného toku zboží a informací od dodavatelů až po konečného zákazníka.

CRM (Customer Relationship Management)

- CRM je komplex technologií, podnikových procesů a personálních zdrojů určených pro řízení a průběžné zajišťování vztahu se zákazníky podniku.
- Vztah se zákazníky je řízen v oblastech podpory obchodních činností, zejména prodeje, marketingu a podpory zákazníka nebo zákaznických služeb.

Customer Relationship Management = řízení vztahů se zákazníky.

CRM - soubor metodik pro řízení a zlepšování vztahů se zákazníky za využití informačních technologií. Jako takové se stává součástí firemní strategie i kultury. Informační systémy se dnes zaměřují na podporu podniku prodat svoje výrobky či služby. Vznikají nové komunikační kanály se zákazníkem. Podniky se pomocí IS snaží být v kontaktu se zákazníkem (klasická pošta, elektronická pošta, diskuse a konference na webu, call centra…). CRM je rozšiřující komponentou ERP II, ale samozřejmě je možné koupit ji i samotnou.

Podstata CRM: udržování a rozvíjení komunikace se zákazníky, snaha vytvářet dojem individuální komunikace. S příchodem SCRM zapojení zákazníka do procesů firmy (reakce na jeho podněty atp).

CRM je komplex technologií (aplikačního a základního software, technických prostředků), podnikových procesů a personálních zdrojů určených pro řízení a průběžné zjišťování vztahů se zákazníky podniku, a to v oblastech podpory obchodních činností, zejména prodeje, marketingu a podpory zákazníka i zákaznických služeb.

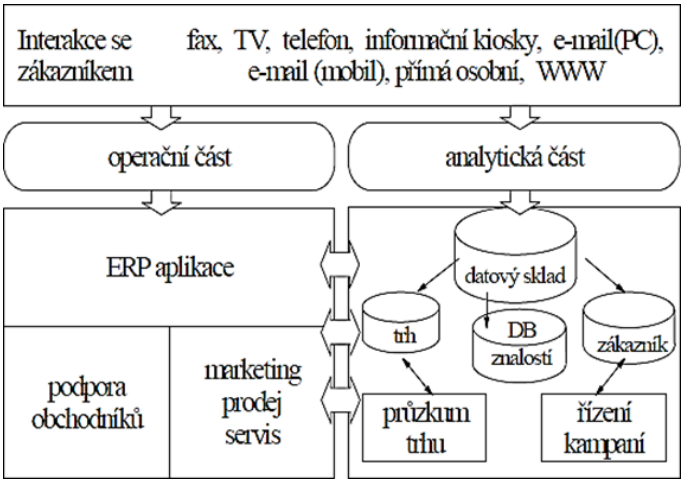
Přínosy CRM:

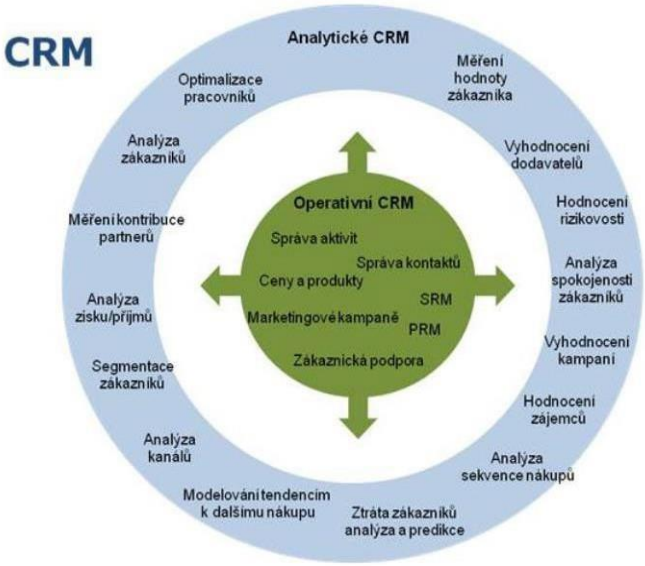
- zlepšení organizace obchodní činnosti firmy, vztahů se zákazníky, zprůhlednění obchodních procesů, jejich přesná evidence, aktualizace a archivace historie
- zdokonalení sledování obchodních příležitostí, sledování a řízení prodejního cyklu u stávajících i nových zákazníků,
- upřesnění odhadů budoucích tržeb, kvalitním a jednoduchým nástrojem na řízení a analýzu obchodu.
- podpora řízení kampaní
- sledování a analýzy chování zákazníků (trhu)

Funkcionalita CRM.

Poskytuje čtyři základní způsoby uplatnění, které v podniku mohou být nasazovány i samostatně:

- Aktivní (active) CRM (pozn.: tohle členění uvádí jen (BASL, 2008))
- Operativní (operational) CRM
- Kooperační (collaborative) CRM
- Analytické (analytical) CRM





BI (Business Intelligence)

- Manažerské systémy jsou využívány pro podporu strategického rozhodování.

- Data pro BI jsou získávána z celé databáze ERP II (především tedy SCM a CRM).
- Nad těmito daty je třeba uvažovat multidimenzionálně (z více úhlů pohledu). Na základě těchto modelů jsou sestavovány statistické ukazatele, které pomáhají při samotném rozhodování.
- Business inteligence (BI) představuje komplex přístupů a aplikací IS/ICT, které téměř výlučně podporují analytické a plánovací činnosti podniků a organizací a jsou postaveny na principu multidimenzionality, kterým zde rozumíme možnost pohlížet na realitu z několika možných úhlů.
- Je to sada procesů, aplikací a technologií, jejich cílem je účinně a účelně podporovat rozhodovací procesy ve firmě.
- BI je orientován na vlastní využití informací v řízení a rozhodování, a nikoli na základní zpracování dat a realizaci běžných obchodní, finančních a další transakcí.
- To, jak jsou možnosti BI využity, dnes do značné míry ovlivňuje výkonnost a kvalitu řízení firmy, a v souvislosti s tím i nakonec její celkovou úspěšnost a konkurenceschopnost.

Význam BI pro podnik

Společnosti uchovávají velké množství dat ze systémů, které používají (ERP, CRM, SCM…). Tato data jsou zpravidla uložena v transakčních databázích (OLTP– OnLine TransactionProcessingdatabáze). Ty jsou navrženy tak, aby umožňovaly ukládání velkého množství dat, jejich snadnou úpravu a mazání. Pokud ale dojde na získávání souhrnných informací, jsou transakční databáze pro tento účel naprosto nevhodné. Dalším problémem je pak také to, že analytická data, která potřebuje management firmy, pocházejí často z různých systémů a jsou tedy uloženy v různých databázích s odlišnou strukturou, což činí jejich získání ještě obtížnějším. Dalším problémem může být neúplnost dat či existence duplicitních záznamů.

Potřebuje-li tedy management firmy, která nepoužívá BI, kompletní a přesné analytické informace pro svá rozhodnutí, tyto informace buď nejsou k dispozici, nebo jejich získání trvá nepřiměřeně dlouhou dobu, což může vést k intuitivnímu rozhodování managementu nepodloženému fakty, které ve výsledku nemusí být vždy správné. Technologie BI dokáží integrovat data ze všech oblastí podnikové činnosti a poskytnout potřebné informace v požadované struktuře.

Princip BI

Základním principem BI je **multidimenzionální pohled na data**. Ten spočívá v možnosti zobrazení požadovaných ukazatelů podle zvolených dimenzí, jinými slovy v poskytování různých úhlů pohleduna danou problematiku. Pro použití BI nástrojů je tedy potřeba mít data uložená v multidimenzionální databázi. Technologie založená na práci s multidimenzionálními databázemi se nazývá OLAP (OnLine AnalyticalProcessing).

Popis fungování

Data jsou proto pro potřeby BI získávána ze zdrojových systémů (např. ERP, CRM, apod.) pomocí tzv. **datových pump** (ETL – Extraction, Transformation andLoading) do datového skladu. **Datový sklad** je speciálně strukturovaná databáze, která obsahuje požadovaná data získaná ze zdrojových systémů. Z datového skladu jsou pak data přetransformována do OLAP databáze (=multidimenzionální databáze), která se využívá pro tvorbu analýz, které pomáhají manažerům při tvorbě nejrůznějších rozhodnutí.

MES

manufacturing execution system - vyrobni informacni proces, tvori vazbu mezi podnikovými informačními systémy a vlastnim vyrobnim systemem - jsou vice specializovane nez ERP

funkce:

- správa výrobních zdrojů - pridelovani zdroju a kapacit
- správa vyrobnich postupu - evidence a sprava verzi a vymena kmenovych dat s okolnimi systemy - vyrobni pravidla, kusovnik, vyr. zdroje
- detailni planovani vyrob - dopredne a zpetne planovani - dle jednoduchych algoritmu X nebo geneticke algoritmy u komplexnich
- dispecerske rizeni - prirazovani prace

- rizeni vyroby - v naplanovane a uvolnene vyrobe - fronta prace - zajistuje aktivity pro rizeni vyroby
- sber dat - historizace procesnich a vyr. dat
- sledovani vyrobku a jejich rodokmen - info o zdrojich - legislativni pozadavky, audity, reklamace
- vykonnostni analyzy - KPI

MES představují návaznost IS na vlastní výrobní systém. Je to v podstatě vrstva mezi ERP, resp. APS, a technologickým procesem, kde mohou být nasazovány různé CNC stroje a zařízení. Tyto systémy zabezpečují detailní sběr dat a jejich zpracování pro účely vyhodnocení výroby a operativního plánování. Ve srovnání s ERP jsou tyto systémy více ovlivněny typem výroby a nejsou tedy tak univerzální jako celopodniková řešení, z čehož plyne jejich větší specializace.

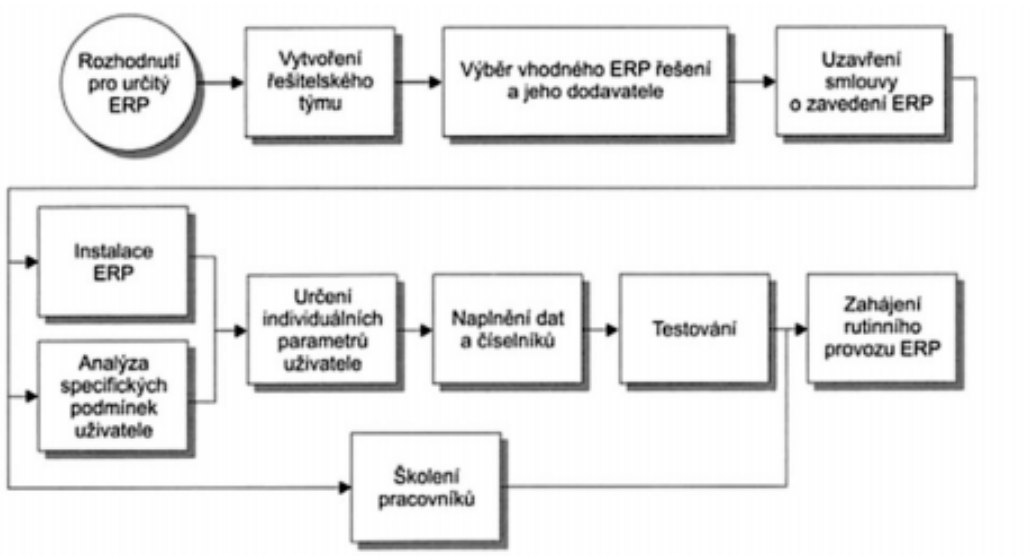
28. Implementace podnikových IS

Projekt nasazení ERP do firmy

Projekt nasazení ERP je soubor činností, jehož počátek lze spatřit ve vyjádření potřeby zavedení nového IS a konec v provozu a údržbě implementovaného IS. Obecně lze v projektu rozlišit 4 etapy:

1. Rozhodnutí pro změnu podnikového IS a vytvoření projektového týmu na straně zákazníka
2. Výběr vhodného řešení
3. Vlastní implementace vybraného řešení
4. Provoz a údržba, případné změnové řízení

Tyto etapy lze vidět i na následujícím obecném schématu implementačních kroků zavádění ERP:



Rozhodnutí pro změnu PIS

Záměr zavést IS vychází z rozhodnutí vedení firmy a musí být v souladu s firemní business strategií. Je provedena důkladná analýza současného stavu v podniku. Dochází k formulaci klíčového problému, který má IS řešit. Vzhledem k tomu, že implementace PIS je náročný proces, je nutné pro tento účel vytvořit vlastní proces.

V této etapě je potřeba vyjasnit si:

- Záměr projektu
- Strategické cíle podniku
- Způsob komunikace s dodavateli, obchodními partnery a zákazníky (ujasnění si, jak v tomto ohledu může nový IS pomoct)
- Současné informační toky v podniku
- Současný stav ICT vybavení podniku (použitý HW, SW, jejich účel a způsob využití;
- dodavatelé, případné současné smluvní závazky)

- Současný stav procesů v podniku
- Potenciál personálu (zkušenosti s ICT, se systémy ERP; slouží zejména k vytvoření
- projektového týmu z kvalifikovaných osob a k odhadu rozsahu nutného zaškolení)
- Finanční priority podniku a uvažovanou částku určenou pro projekt.

Formulují se **potřeby**, které má IS naplnit – může definovat očekávaný stav po zavedení systému. Jsou nalezeny explicitní cíle a jejich metriky. Pokud dojde k rozhodnutí, že se nový IS bude zavádět, je složen **projektový tým**, který bude primárním subjektem spolupráce s dodavatelem systému. Pokud je potřeba, je projektový tým vzděláván v řešené problematice. Jsou stanovena **výběrová kritéria** pro systém a dodavatele, je vytvořen **projektový plán** a celkovým produktem první fáze je **poptávkový dokument a zadávací dokumentace**, které mohou být rozeslány potenciálním dodavatelům.

Výběr vhodného řešení (výběrové řízení)

S ohledem na potřeby a finanční možnosti firmy dochází k analýze trhu a výběru vhodného IS a dodavatele. Výrazně v tomto kroku mohou pomoct:

Demoverze systémů – lze si vyzkoušet práci s ukázkovými, případně i vlastními daty

Návštěvy referenčních instalací, schůzky s dodavateli

Hodnotící kritéria pro zkoumání variant jsou podrobněji popsána v rámci otázky 44 – Hlavní kritéria výběru ERP. Kritéria jsou vyhodnocena a dojde k výběru jednoho či více alternativních dodavatelů a IS. Uvažovaným dodavatelům je zaslán poptávkový dokument s požadavky a cíli, jejichž splnění zákazník žádá. Tyto cíle, požadavky a také relevantní závěry z analýzy současného stavu ve firmě, je v dokumentu vhodné identifikovat co nejpřesněji, aby byla snížena případná míra nepřesnosti nabídky od dodavatelů.

V rámci výběrového řízení jsou vyjednávány obchodní podmínky a na jeho konci je podepsána smlouva s dodavatelem.

Někdy může být součástí této etapy, ještě před podepsáním smlouvy, vypracování úvodní studie dodavatelem, v jiných případech může být zpracována až po uzavření smlouvy již jako součást etapy implementační. Tato studie je zpracována na základě informací poskytnutých zákazníkem (zadávací dokumentace, výstup analýzy z první etapy) a obsahuje např. následující:

1. **Shrnutí požadavků zákazníka** – základní cíle, organizační cíle (jak systém ovlivní organizační strukturu firmy), technické cíle (přínosy systému v technických oblastech provozu), aplikační cíle (pokrytí funkčních požadavků)
2. **Popis současného stavu** – stávající organizační struktura, současné zabezpečované funkce, současné toky dat a informací (popis vazeb mezi funkcemi)
3. **Standardní funkční struktura zaváděných modulů ERP** – popis implementovaných funkčních oblastí; soupis modulů a jejich funkcí
4. **Popis HW platformy** – popis potřebného HW vybavení potřebného pro provoz systému; struktura sítě apod.
5. **Technologické prostředí systému ERP** – koncepce serverů, systémového SW, databází, případně dalšího aplikačního SW; koncepce uživatelských oprávnění, rozhraní pro přenos dat, filosofie tvorby rozhraní, řešení zálohování a archivace, popis případného upgradu systému, zajištění provozu při výpadku centrálního serveru (např. použití náhradního); popis technologické podpory nabízené dodavatelem
6. **Koncept implementace jednotlivých funkčních oblastí:**
    - Pokrytí podnikových procesů funkcemi ERP (mapování funkcí a modulů ERP na procesy); popis funkcí a zaváděných modulů
    - Návrh na způsob pokrytí případných částí PIS, které nejsou řešené systémem ERP (seznam ostatních aplikací a jejich integrace s ERP; může jít i o dočasné řešení, pokud je ERP nasazován po částech a vytěsňuje nějaký stávající IS podniku)
    - Data – výčet dat, která budou spravována přímo systémem a dat, která budou spravována ostatními aplikacemi, ale mohou být systémem také využívána, či dat ze starého systému, která budou převedena do nového

    - Rozhraní – např. definice rozhraní pro převod dat starého systému do nového či rozhraní, která budou zajišťovat přenos dat mezi ERP a jinými aplikacemi
    - Podklady pro nastavení systému – popis základních definovaných prvků pro nastavení; poskytuje základ pro pozdější detailní nastavení
    - Nestandardní rozšíření systému – popis funkcionality, kterou požaduje zákazník, ale nelze ji realizovat pomocí ERP; dojde např. k jejímu dovývoji
    - Návrh dalšího rozvoje ERP (zejména pokud bude nasazován postupně)
    - Organizační struktura (může jít o výčet členů řešitelského týmu, ale i o případné pozměňovací návrhy na organizační strukturu zákaznické firmy, která bude korespondovat s použitím systému)
7. **Integrační vazby** – popis vazeb mezi funkčními oblastmi ERP (moduly) a vazeb směřujících mimo ERP (integrace s vnějšími aplikacemi)
8. **Časový harmonogram implementace ERP**

Vlastní implementace vybraného ERP

Po podpisu smlouvy započnou přípravy na implementaci systému a následně samotná implementace. Dodavatelé se při implementaci zpravidla řídí nějakou metodikou (může jít o vlastní firemní metodiku či nějakou standardní metodiku; jako příklad použitelných metodik lze zmínit Microsoft Dynamics SureStep, metodika MMDIS).

Na počátku implementace **je přípravná fáze**, kdy dochází zpravidla k:

- **Analýze požadavků a návrhu koncepce řešení** – v zásadě odpovídá popisu úvodní studie z předchozí části. Buďto tedy k analýze dojde teď anebo již byla provedena. (Pozn.: i některé další body mohou být součástí již případně proběhlé úvodní studie)
- **Stanovení pravidel organizace a komunikace** v rámci projektového týmu mezi dodavatelem a zákaznickou firmou; plánování schůzek dohlížecího výboru (dohlíží zástupci z řady vedení zákaznické firmy i dodavatelské firmy)
- **Instalace ERP systému**, včetně eventuální dodávky potřebného HW a základního SW
- **Zaškolení osob** – přehledové školení pro manažery, školení projektového týmu v problematice implementace, školení koncových uživatelů, případně školení IT specialistů, kteří budou zajišťovat provoz systému
- **Stanovení organizace toku dat,** určení zodpovědností za data
- **Specifikace a nastavení důležitých parametrů ERP**, pomocí nichž se ERP customizuje na konkrétní podmínky podniku
- **Analýza podnikových procesů** a jejich korelace s procesy v referenčních modelech ERP systému
- **Návrh formulářů pro komunikaci se systémem** – jak formuláře pro zaměstnance podniku, tak formuláře pro obchodní partnery či zákazníky
- **Stanovení způsobu převedení provozu ze současného systému na nový** (pokud je v podniku nějaký stávající systém) – řešení mohou být různá – jednorázové překlopení do nového systému, postupné nahrazování funkčních oblastí, paralelní provoz, různé kombinace…
- **Celkově dochází k detailní analýze** – HW, SW, dat, procesů, organizační struktury…

Pro realizaci implementace vytváří dodavatel společně se zákazníkem její detailní návrh, přičemž jde zejména o:

- Přesný způsob nasazení a nastavení systému
- **Návrh databáze**, příprava dat, která budou uložena ještě před spuštěním systému
- Specifikace a naplnění číselníků
- **Realizace integrace ERP s ostatními aplikacemi podniku**; přesné určení datových rozhraní (pro datovou komunikaci mezi ERP a ostatními aplikacemi)
- Specifikace funkcí, které je třeba dovyvinout

Systém je následně customizován a chybějící funkcionalita dle potřeby dovyvíjena.

V závěru implementační fáze jsou spouštěny jednotlivé moduly systému, je dokončeno veškeré nastavení nutné pro požadovaný chod systému a je dodána uživatelská dokumentace. Než dojde k ostrému provozu systému, je nejprve

testován (zkušební provoz na testovací sadě dat) a podroben akceptačnímu řízení. Výsledkem této etapy je zahájení provozu IS, který bude následně používán, udržován a případně rozvíjen.

Náklady spojené se zavedením IS

- Cena IS se skládá ze dvou částí:
    - Jednorázové náklady
    - Nákup hardwaru
    - Nákup softwaru – obvykle licencí
    - Datové naplnění systému a tvorba datových rozhraní na existující řešení v podniku
    - Úpravy obrazovek a sestav, tvorba a tisk nových formulářů
    - Doprogramování speciálních úloh
    - Úpravy podnikových procesů
    - Školení
- Provozní náklady
    - Servisní poplatky za hardware
    - Servisní poplatky za software
    - Poradenská činnost
    - Zabezpečení provozu vlastního IT oddělení

Etapy fáze implementace

Zhruba odpovídá metodice MMDIS.

Úvodní studie

Formuluje základní požadavky na IS/IT v rámci projektu. Definuje základní koncept navrhovaného řešení a návaznosti na ostatní projekty. Jedním z podstatných cílů je určit rozsah úprav aplikačního software vzhledem k potřebám zákazníka.

Co obsahuje úvodní studie:

- Formulace **cílů** projektu
- Analýza současného stavu IS/IT
- Definovat **klíčové požadavky** na IS/IT
- Návrh celkového **konceptu** řešení
- Návrh aplikačního software
- Návrh základního software
- Specifikace vlivů projektů na organizaci podniku
- Návrh způsobu řízení projektu
- Rámcový návrh způsobu **migrace** na nové řešení
- Specifikace poskytovaných **doprovodných služeb** dodavatelem
- Návrh **harmonogramu** řešení
- Ekonomický rozbor projektu

Globální analýza a návrh

Fáze zahrnující globální analýzu a návrh **obsahu a způsobu řešení**, včetně hrubé **specifikace funkcí**.

- Podrobná specifikace **organizace projektových činností**
- **Školení** pracovních týmů
- Analýza uživatelských požadavků
- Analýza **funkčních požadavků uživatelů**, vzhledem k možnostem ASW
- Vytvoření harmonogramu

Detailní analýza a návrh

Fáze zahrnující **detailní analýzu a návrh IS**, **jednotlivých databází, programových modulů, řídicích procedur, organizace** apod.

- Prototypování
- Návrh datových rozhraní
- Návrh změn organizačních struktur
- Návrh kmenových databází
- Návrh vstupních informací

Customizace/vývoj

Customizace přestavuje vytvoření nebo úpravu programu podle požadavků zákazníka.

- Vytvoření potřebných úrovní dokumentace, specifikaci nároků na konverzi dat apod.
- Nastavení parametrů modulů
- Realizace datových rozhraní
- Vytvoření uživatelské dokumentace

Migrace, zavedení do provozu

Zavedení projektu do provozu a migrace systému zahrnuje **provozní a organizační opatření**.

- **Instalace** customizovaných modulů a potřebného SW a HW
- Zajištění správy systému
- Zajištění **školení**
- Konverze dat do provozního systému
- Zajištění integračních testů
- Analýza **zatížení** systému

Provoz, změnová řízení

Tato etapa zahrnuje **běžné údržbové operace**, provozní servis a permanentní **konzultační služby** – helpdesk.

- Monitorování a optimalizace řešení operací IS/IT
- Analýza organizačních struktur
- Vyhodnocování **nových požadavků** na IS/IT a provádění **dílčích úprav**
- Nasazování aplikačních software vyšších verzí – **upgrade systému**

29. Přínosy, rizika a trendy podnikových IS

Tendence v oblasti PIS

Zpočátku byly komplexní PIS výsadou velkých podniků, v posledních letech se rozšiřují i do středních a malých podniků (dotace, nové přístupy vhodnější pro tento typ zákazníků...)

- **Doba trvání implementace IS se zkracuje**
- Údržba IS se stává nákladnější
- Vetší důraz na služby spojené s dodávaným IS
- Nové formy řešení PIS – **SOA (SaaS), cloud, open source** ...
- Dříve si dodavatelé PIS kladli otázku „Jak co nejlépe naimplementovat systém dle požadavků klienta“, dnes je to otázka „Jak podpořit konkurenční výhodu svého partnera, aby byl „dlouhodobě úspěšný“.
- Velké **množství akvizic** na trhu – velké firmy skupují menší a rozšiřují tak své portfolio

	Dříve	Nyní
Vývoj podnikových aplikací	• IS vyvíjeny často jako pilotní aplikace pro nové zákazníky • Nové funkcionality IS zvyšují potenciál produktu na trhu	• IS rozvíjeny již jako řešení pro skupinu stávajících zákazníků • IS začínají zahrnovat i nezanedbatelné množství nadbytečného kódu
Implementace podnikových aplikací	• Implementace IS prováděny víceméně stylem budování na „zelené louce“ • Důležité je zprovoznění aplikace • Přínosem implementace IS jsou změny podnikových procesů a uspořádání podnikových dat	• Implementace IS musejí více respektovat stávající mozaiku aplikací • Roste důležitost požadavků na snadný a nenákladný provoz • Prováděné inovace IS mohou být limitovány technologickými, organizačními a znalostními tendencemi ke konzervaci stávajícího stavu procesů, dat a integrovaných aplikací
Údržba podnikových aplikací	• Zdroje a úsilí je věnováno rozvoji nových IS a údržba chápána spíše jako etapa následující po zavedení informačního systému	• Údržba podnikových IS vyžaduje vyšší spotřebu zdrojů • Roste potřeba optimalizace služeb poskytovaných podnikovým IS
Uživatelé podnikových aplikací	• Pro provedení implementace a zajištění obsluhy IS je potřebná značná osvěta a školení uživatelů i managementu	• Uživatelé jsou již znalí ovládání IS, mnohdy si však stále zachovávají původní návyky a chování osvojené před zavedením IS

Trendy ve zlepšování nabídky ERP

Orientace na malé a střední podniky

Dříve byly systémy ERP používány především ve větších podnicích a organizacích. Ty jsou ale těmito produkty v současnosti takřka nasyceny. Dříve bylo možné dobře identifikovat řešení ERP vhodná pro vetší podniky a vedle nich i řešení pro střední a malé podniky, ale v současné době toto dělení postupně přestává platit a oblast malých a středních podniků se stává cílovou skupinou všech dodavatelů ERP.

Řešení pro **malé a střední podniky** však není jen jakýmsi „zmenšením“ funkčních možnosti, ale vyznačuje se i osobitými metodami implementace a charakteristickým přístupem konzultantů dodavatelů. Mezi specifické požadavky menších podniků patří nižší cena, kratší doba implementace a vetší tlak na přínosy zavedení IS.

Malé a střední podniky také využívají možnost **získání dotací na pořízení PIS**. Tím je však trh deformován, neboť dotace vedou k často lehkovážnému způsobu výběru dodavatele, a tedy i prosazení méně kvalitních řešení podpořených marketingovými kampaněmi.

Zpracování dat i ze zdrojů mimo podnik

Standardně jsou v ERP dominantně využívána vnitropodniková data, především data finanční a logisticko-výrobní. Postupně ale dochází k rozšiřování o **data z okolí podniku** související s dodávkami (a dodavateli) a se zákazníky. Ještě stále však tyto systémy v nedostatečné míře integrují data o konkurenci, která jsou nutná k inovacím, a také data o výzkumu a vývoji v daném oboru.

Směrování k integraci a mobilní komunikaci

K hlavním trendům, které ovlivňují a budou ovlivňovat oblast ERP, patří **integrace systémů na bázi XML** (systémy si vyměňují data prostřednictvím XML dokumentů). Směr vývoje pokračuje k integraci dalších objektů, resp. Dokumentů nejrůznějších forem. K významným změnám dochází v oblasti přenosu dat. Dřívější tištené výstupy a sestavy

nahradily údaje z databáze zobrazované v režimu on-line, a to prostřednictvím koncových terminálů a osobních počítačů. Dnes se tato zařízení rozšiřují o **nové formy komunikace**, jako jsou např. mobil

Optimalizační a simulační metody

Dosud nejčastěji používaná metoda MRPII, resp. ERP, je stále více doplňována o podporu rozhodování na bázi optimalizace, simulace a analýzy what-if. Jsou používány nové algoritmy, nové možnosti informační techniky, jako jsou vetší paměti a rychlé procesory. Druhý významný směr v oblasti metod je příklon k řízení podle principů projektového řízení.

Uživatelé PIS

Dříve byla častým problémem **rezistence zaměstnanců vůči zavádění nových systémů a procesů** – management podceňoval nutnost důkladného školení uživatelů a v důsledku toho zaměstnanci nový systém nevyužívali optimálním způsobem, či se jeho použití dokonce ze strachu snažili vyhýbat. V současné době se tento jev podařilo úspěšně minimalizovat, neboť se věnuje vetší pozornost školení uživatelů, dále jsou k dispozici uživatelské příručky, popř. i helpdesk. Navíc jsou systémy intuitivnější a uživatelsky přívětivější.

Nové aplikace

V současnosti jsou kandidáty pro začleňování do ERP systému aplikace PDM (Product Data Management) a MES (Manufacturing Execution Systems).

SOA – Service Oriented Architecture

Mezi nejvýznamnější trendy současných PIS patří **Service Oriented Architecture (SaaS – Software as a Service,** také nazýván jako přístup **on-demand**, je to forma outsourcingu). Hlavní myšlenkou je poskytování některých součástí PIS formou služby, kdy jsou aplikace nainstalovány a provozovány na serverech dodavatele a zákazníkům jsou k dispozici prostřednictvím internetu. Podnik tedy neplatí licenci za všechny programy (tedy i ty nepotřebné), ale platí pouze za to, co využívá. Vzhledem k tomu, že jde o standardizovaná řešení, je tato možnost obvykle levná, protože se náklady rozloží mezi více podniků. Navíc podniku odpadá starost se správou serverů a aplikací, aktualizacemi, podporou uživatelů atd., neboť se tato odpovědnost přesouvá na stranu dodavatele. Nejvýznamnější firmou, která své podnikání zcela postavila na SaaS, je společnost **Salesforce**.

Cloud

Cloud je technologie umožňující poskytovat IT infrastrukturu a software jako službu. Obvykle jde o standardizovaný software s možností drobné parametrizace, který neumožnuje velkou míru individuality (řešení by pak bylo příliš nákladné). Cloud je vhodný pro firmy, které nemají vlastní kvalitní IT pracovníky a nezvládají řízení svých IT služeb. Využívají ho v současnosti spíše menší, dynamicky se rozvíjející firmy, které nemají problém se přizpůsobit standardům daného systému.

Cloud se skládá ze 3 částí:

- **IaaS (Infrastructure as a Service)** – infrastruktura (tj. servery, storage řešení, zálohovací řešení, firewally, atd.)
- **PaaS (Platform as a Service)** – vývojová a aplikační platforma (zejména operační systém, databázový server, atd.
- **SaaS (Software as a Service)** – software (poštovní server, CRM systém, ERP systém, kancelářské aplikace atd.)

Všechny tyto tři prvky pak poskytovatel cloudového řešení poskytuje svým zákazníkům, a to obvykle prostřednictvím internetu nebo jiné vysokorychlostní datové sítě. Poskytovatel Cloud computingu provozuje infrastrukturu, stará se o její údržbu, obměnu, napájení atd. Na své infrastruktuře pak provozuje případně i dané aplikace a operační systémy, které opět spravuje, záplatuje, v případě potřeby reinstaluje, upgraduje na nové verze atd. Tuto infrastrukturu pak sdílí pro desítky, stovky až tisíce zákazníků.

Zákazníci si od poskytovatele Cloud computingu vyberou nějakou službu, kterou pak využívají vzdáleně (služba běží na infrastruktuře poskytovatele) prostřednictvím klientské aplikace (velice často i webového prohlížeče). Zákazník tedy nic neinstaluje, nic nekonfiguruje (vyjma klientské aplikace), nic neudržuje, nic neupgraduje, a dokonce do

ničeho neinvestuje. Platí jen pravidelné měsíční poplatky za službu, které se obvykle odvíjejí od počtu licencí (uživatelů služby) a/nebo od její kapacity (velikost úložného prostoru, pronajatý výkon atd.). Mezi běžné využívané služby cloudu patří např. emailové klienty (gmail, seznam.cz...), uloz.to, skype...

Open source

Open source vzniká v prostředí neformálního seskupení vývojářů, kteří mají zájem na vzniku dané aplikace. Výhodou otevřené práce v komunitě je sdílení duševních schopností širšího okruhu expertů, než je možné v rámci vývojářského týmu ve firmě. Open source software (OSS) je k dispozici zdarma a klade důraz na otevřenost a možnost libovolných úprav. Výhodou je také standardizace, u déletrvajících projektů pak lze očekávat stabilní chod a optimální výkon (na kód dohlíží mnohem větší počet lidí než při klasickém vývoji a navzájem se kontrolují).

Nevýhodu lze vidět v tom, že na OSS neexistuje ze strany vyvíjející komunity žádná garance. Naštěstí je už ale mnoho OSS poskytováno důvěryhodnými dodavateli, kteří jsou schopni zaručit správnou funkcionalitu softwaru, opravy chyb a další vývoj. Navíc se postará o integraci softwaru do stávajícího informačního systému podniku, zajistí školení, podporu atd.

Pokud si podnik pořídí OSS od dodavatele, jsou mu účtovány jen poplatky za služby s ním spojené (licence se neplatí) – doprogramování, údržba, podpora, školení uživatelů atd

30. Propojení IT a businessu

Metodika MMDIS

MMDIS (Multidimensional Management and Development of Information System) metodika si klade za cíl přispívat k výchově ICT specialistů, kteří zvýší úspěšnost ICT projektů.

Cílem je **vývoj, údržba a provoz komplexního a integrovaného informačního systému**, který optimálně využívá potenciálu dostupných informačních a komunikačních technologií (ICT) k maximální podpoře podnikových cílů

Komplexní IS je takový, který podporuje všechny činnosti podniku, pro které je možné nalézt efektivní informatickou podporu. Integrovaný IS znamená, že IS je tvořen z celé řady HW, SW a datových komponent, které jsou navzájem propojeny do jediného systému. To, že optimálně využívá potencionálu dostupných ICT, znamená, že není nutně postaven na nejnovějších technologických ICT službách, ale vybírá z nich ty, které mají pro daný podnikový IS ekonomický smysl.

MMDIS zajišťuje **efektivní podporu podnikových cílů a podnikových procesů** pomocí integrovaných informatických služeb, procesů a zdrojů a zdrojů.

MMDIS je globální metodika. Skládá se z 11 základních principů řízení a 5 navzájem propojených konceptuálních modelů (konceptů) řízení.

Charakteristika

- Fáze: GST, IST, US, GAN, DAN, IM, ZA, PU a VY
- Dimenze: HW, SW, INF, PRO, UI, PRA, ORG, BZP, EKO
- Role: top manažer, vlastník business procesu, uživatel, analytik, architekt, vývojář, implementátor

Principy

- myšlenkový přístup k chápání a analýze problému a s ním spojené zásady (pravidla) řešení problému
- principy MMDIS jsou aplikovány v konceptech MMDIS
- principy jsou základem pro metody, techniky a nástroje řízení

Pravidla multidimenzionality

každý složitý problém je nutné analyzovat, hodnotit a jeho řešení navrhovat z mnoha dimenzí (pohledů).

Pravidla:

1. identifikuj všechny dimenze ovlivňující řešení problému
2. vyřeš problém nejdříve z pohledu každé definované dimenze
3. integruj separátní řešení do výsledného řešení (kombinace dimenzí a optimalizace)

11 základních principů metodiky MMDIS

- multidimenzionalita
- vrstvenost
- integrace
- flexibilita
- otevřenost
- standardizace
- kooperace
- procesní pojetí
- učení a růstu (postupné zlepšování procesů)
- lokalizace zdrojů a rozhodnutí
- měřitelnost (metriky)

5 základních konceptuálních modelů metodiky MMDIS

- model procesně řízené organizace
- integrace strategie, procesů, služeb a zdrojů (model SPSR)
- integrace oblastí řízení (model systémové integrace)
- model tvorby a dalšího rozvoje IS
- systém řízení IS (model řízení informatických procesů ve firmě)

3 skupiny pohledů

Pro řešení informačního systému MMDIS doporučuje použít pohledy zařazené do 3 skupin:

První skupina pohledů

Je reprezentována dvěma pohledy:

- **uživatelským** – odpoví na otázku, komu je IS určen a jaké ICT služby bude nabízet jednotlivým skupinám uživatelů
- **řešitelským** – odpoví na otázku, jak požadované systém vytvoříme, jak ho budeme provozovat a jak budeme ICT služby dodávat

Druhá skupina pohledů

Reprezentuje úrovně integrace IS/ICT, použitou úroveň abstrakce a časovou dimenzi řešení. Tato skupina dimenzí se v MMDIS promítá do jednotlivých procesů a fází vývoje IS/ICT.

Třetí skupina pohledů

Zahrnuje ty dimenze IS/ICT, které se aplikují v každé fázi vývoje IS/ICT. Jedná se o tyto „obsahové dimenze IS/ICT“: (inf)informační/datová, (pro) procesní/funkční, (eko) ekonomická/finanční, (org) organizační/legislativní, (pra) pracovní/sociální/etická, (ui) uživatelský interface, (bzp) bezpečnost a kvalita, (sw) softwarová, (hw) hardwarová, (met) metodická, (dok) dokumentační, (mng) manažerská

Fáze

- Globální podniková strategie (GST)

- Informační strategie (IST)
- Úvodní studie (US)
- Globální analýza a návrh (GAN)
- Detailní analýza a návrh (DAN)
- Implementace (IM)
- Zavádění systému (ZA)
- Provoz a údržba (PU)
- Vyřazení systému (VY)

Role

- Top manažer
- Vlastník business procesu
- Uživatel
- Analytik
- Architekt
- Vývojář
- Implementátor
- Manažer IS/ICT (CIO)
- Správce
- Konzultant

Dimenze

1. Data/informace (inf)

Typy informací, které jsou zapotřebí při jednotlivých podnikových aktivitách, obsah a struktura datové základy podniku a její fyzická realizace. Řešením této dimenze v jednotlivých fázích vývoje IS/ICT je navrhována a realizována datová architektura IS/ICT. Vazba na dimenzi **pro**, jelikož jednotlivá data reprezentují mimo jiné i vstupy a výstupy ve specifických procesech.

2. Funkce/procesy (pro)

Procesy probíhající v podniku a možnost jejich podpory informačním systémem. Řešením této dimenze v jednotlivých fázích vývoje IS/ICT je navrhována a realizována funkční a procesní architektura IS/ICT

3. Organizační a legislativní aspekty (org)

Platná legislativa země, ve které podnik působí, podnikové směrnice a normy, platné obecné standardy a normy (např. ISO, ČSN), vliv zavedení nového IS/ICT na organizaci podniku. Náplní je také definování pravidel pro vývoj, údržbu a provoz IS/CIT. Definují se útvary zaměřené na zpracování dat, zodpovědnosti a pravomoci jejich pracovníků, postupy prevence proti mimořádným událostem atd. Vazba na **inf i pro** – data musí nějak vypadat a musí se s nimi pracovat konkrétním způsobem, tj. s určitým omezením.

4. Pracovní, sociální a etické aspekty (pra)

Potřebná struktura pracovníků podniku (počet, kvalifikace, věk, pohlaví) pro stav po zavedení nové verze IS/ICT, možné sociální a etické dopady přechodu na novou verzi IS/ICT, potřebná opatření (nábor, resp. propouštění pracovníků, rekvalifikace a školení pracovníků) zavedení nové verze IS/ICT. Vazba např. na **pro** – jakým způsobem bude probíhat proces školení či přijímání nového zaměstnance.

5. Software (sw)

Určení architektury programového systému, tj. určení typů, parametrů a vzájemných vazeb jednotlivých komponent základního a aplikačního programového vybavení a jednotlivých programových modulů. Určení, zda bude daná softwarová komponenta nakoupena, resp. vyvinuta vlastními silami. Vazba je možná na cokoliv, jen to dobře odůvodnit. Např. vztah k **eko** – řešení přínosu dalšího SW vzhledem k jeho nákladům.

6. Hardware (hw)

Hardwarová architektura IS/ICT, tj. typy, parametry a počty počítačů, periferních zařízení, komunikačních sítí a dalších technických prostředků. Vazba je možná na cokoliv, jen to dobře odůvodnit. Např. vztah k **eko** – řešení přínosu dalšího HW vzhledem k jeho nákladům.

7. Ekonomické a finanční aspekty (eko)

Finanční náklady tvorby a provozu IS/ICT a přínosy podniku z užití IS/ICT. V jakém období a z jakých zdrojů budou náklady na IS/ICT kryty.

8. Metody (met)

Metody a s nimi související techniky a nástroje používané v jednotlivých fázích vývoje IS/ICT pro analýzu, návrh, implementaci a provoz IS/ICT.

9. Dokumenty (dok)

Jaké dokumenty vznikají v průběhu řešení a provozu IS, jaký mají obsah a jak na sebe navzájem navazují.

10. Řízení prací dané fáze (mng)

Podklady k optimalizaci spotřeby lidských, finančních, materiálových a dalších zdrojů tak, aby se s minimálními nároky na kapacitu těchto zdrojů dosáhlo v plánovaném čase požadovaných cílů IS/ICT.

Váhy dimenzí

Váha, resp. významnost téže dimenze se v jednotlivých fázích mění. Např. organizační a ekonomická dimenze mají podstatně větší váhu v úvodních fázích (GST, IST, US), kdežto SW a HW dimenze mají naopak větší váhu v závěrečných fázích (GAN, DAN, IM) řešení IS/ICT.

Vazby mezi obsahovými dimenzemi

Konzistentní řešení => potřeba řešit jednotlivé dimenze ve vzájemné závislosti => nutná analýza a návrh všech vztahů

31. Modelování business procesů

K modelování procesu existuje řada různých přístupů a norem, vzniklých různými způsoby a zdůrazňujících různé aspekty procesu, jakož i různé aspekty ignorujících.

Základní prvky modelu:

- **Proces** – posloupnost činností jako reakce na událost
- **Činnost** – základní prvek procesu, má vstupy a výstupy
- **Událost** – externí podnět procesu/činnosti
- **Stav** – výsledek nějaké činnosti v systému

Proces

„Účelně naplánovaná a realizovaná posloupnost činností, ve kterých za pomoci odpovídajících zdrojů probíhá transformace vstupů na výstupy.“ „Soubor činností, který vyžaduje jeden nebo více vstupů a tvoří výstup, který má nějakou hodnotu pro zákazníka.“

Obecně – reakce systému na určitou událost (časovou, datovou, mimořádnou). Procesů je celá řada, setkáváme se s nimi v osobním i profesním životě každý den a jsme jejich součástí. K modelování se nejčastěji využívají podnikové procesy.

Podnikový proces – je uvažován jako sled či posloupnost činností podniku, které na sebe navzájem navazují a vedou k dosažení určitého cíle (nejčastěji tvorba hodnoty pro zákazníka)

Charakteristika procesu:

- iniciován událostí
- Výstupem je událost signalizující dosažení jednoho z koncových stavů procesu
- Cílem procesu je přidaná hodnota pro zákazníka

Další komponenty

- **Seznam procesů:**
  - Seznam procesů by měl obsahovat názvy všech procesů, které jsou součástí analýzy nebo modelování. Můžete uvést další informace, jako jsou odpovědné osoby nebo oddělení.
- **Model návaznosti procesů:**
  - Popište, jak jednotlivé procesy navazují na sebe. Tento model by měl obsahovat grafické znázornění jednotlivých procesů a spojení mezi nimi, aby bylo jasné, jakým způsobem se procesy propojují.
- **Základní popisná tabulka:**
  - Pro každý proces vytvořte tabulku obsahující základní charakteristiky, jako jsou název procesu, popis činností, odpovědné osoby, klíčové termíny a další relevantní informace.
- **Model průběhu procesu:**
  - Tento model by měl podrobně popsat průběh jednotlivých procesů, včetně vstupů, výstupů, časových rámů, kritických milníků a dalších relevantních aspektů.
- **Tabulka pro popis konzistencí:**
  - Tato tabulka by měla zachycovat konzistenci a návaznosti na další dimenze modelování systému, například na technickou architekturu, datové toky nebo podnikové strategie.

Procesní modelování

Cíle procesního modelování

Simulace reálných procesů, které probíhají v organizaci

Zachytit chování reality podnikových procesů

Účel procesního modelování

- Optimalizace podnikových procesů
- Integrace aplikací a obchodní procesů
- Proklad pro analýzy dopadu
- Dokumentace systému jakosti – ISO certifikace
- Oddělení know-how od lidí
- Dokumentace pracovních postupů
- Nalezení úzkých míst organizace
- Podpora při návrhu IS a ASW
- Popis podnikového Workflow
- Zjednodušení školení nových pracovníků

Principy procesního modelování

- Vhodné pojmenování
- Dodržení grafické notace

Pro procesní modelování se využívá například notace **BPMN, EPC**

EPC

Metoda Event-driven Process Chain zahrnuje EPC diagram, respektive jeho rozšířenou verzi eEPC diagram (rozšířený o informační toky a organizační jednotky, které s procesem souvisejí), tedy diagram řízený událostmi. Popisuje nejčastěji podnikové procesy a pracovní postupy. Diagram se skládá z prvků:

- **Aktivity** – jsou základem pro modelování procesu a díky nim dosahuje svého cíle. Aktivitou může být např. posouzení objednávky
- **Události** – propojují jednotlivé aktivity. Událostí může být např. došlá objednávka
- **Logické spojky** – používají se ke spojení předchozích dvou elementů. Nabývají hodnot OR (V), AND (^) a exklusive OR (XOR).
- **Role**
- **Datové zdroje**

BPMN

Business Process Model and Notation je soubor pravidel a principů, který slouží pro grafické znázornění podnikových procesů. Zahrnuje procesní diagram, který obsahuje následující prvky:

Flow objects - Tokové objekty

- Events - Události
- Activities - Aktivity / Činnosti

Gateways - Brány

- Connecting objects - Spojovací objekty
- Sequence flow - Sekvenční tok
- Message flow - Tok zpráv
- Association - Asociace

Swim lanes - Plavecké dráhy (Kontexty)

- Pool - Bazény (Kontext)
- Lane - Dráhy (Oddíly kontextu)

Artifacts - Artefakty

- Data object - Datové Objekty
- Group - Skupiny
- Annotation – Anotace

Zralost procesů = Capability Maturity Model (CMM) - Strukturovaný soubor praktik, jejichž aplikace vede ke zvyšování efektivity procesu

Metoda KBPR (Knowledge Based Process Reengineering)

Východiskem metody je kritika "mechanických" přístupů k mapování a reengineeringu podnikových procesů, které jsou příliš technicky zaměřené, málo respektují znalosti lidí podílejících se na procesech a nedostatečně zohledňují nutnou míru kreativity člověka v příslušném procesu.

Nástroje k modelování procesů a funkcí

- Notace modelování procesů BPMN, ARIS (EPC)
- Nástroj pro modelování od Sybase – PowerDesigner
- Univerzální OO modelovací jazyk – UML – např. diagram případů užití, diagram aktivit
- Modelování funkcí pomocí DFD – Data Flow Diagram

32. Kryptografie, kryptografická primitiva a protokoly, šifrování, asymetrická kryptografie

Kryptografie je věda zabývající se důvěrností komunikace přes nezabezpečený komunikační kanál, zabývá se způsoby jak zašifrovat odesílaná data tím způsobem, aby jim nepovolaná osoba nemohla porozumět.

Historie: Celé období kryptografie můžeme rozdělit do dvou částí. Tou první je klasická kryptografie, která přibližně trvala do poloviny 20. stol. První část se vyznačovala tím, že k šifrování stačila pouze tužka a papír, případně jiné jednoduché pomůcky. Během 1. poloviny 20. stol. ale začaly vznikat různé sofistikované přístroje, které umožňovaly složitější postup při šifrování. Tím přibližně začala druhá část, kterou nazýváme moderní kryptografie.

Kryptoanalýza na rozdíl od kryptografie se zabývá právě slabinami šifrovacích systémů a způsoby, jak dešifrovat zašifrované zprávy bez jakékoliv znalosti tajných klíčů.

Historie, od tajného předávání zpráv ve starověku, přes neviditelný inkoust, permutační šifry a enigmu až k současnosti je vlastně neustálý boj kryptografie a kryptoanalýzy mezi sebou.

Kryptologie = Kryptografie + kryptoanalýza

Cíle kryptografie

- **Důvěrnost (data confidentiality)** – udržení obsahu zprávy v tajnosti, omezení počtu osob které můžou vidět obsah komunikace.
- **Autentizace (authentication)** – prokázání totožnosti, tj. ověření, že ten, s kým komunikujeme, je ve skutečnosti ten, s kým si myslíme, že komunikujeme.
- **Integrita (data integrity)** – Zamezení neoprávněné modifikace dat (např. smazání části dat, vložení nových nebo úprava již stávajících dat), ověření, že se zpráva při přenosu nezměnila.
- **Neodmítnutelnost (non-repudiation)** – Jistota, že autor nemůže popřít autorství své zprávy (příjemce nemůže zapřít příjem zprávy, odesílatel odeslání)

Kryptografická primitiva – algoritmy se základními kryptografickými vlastnostmi, ze kterých se skládají kryptografické protokoly.

Kryptografické protokoly

Kryptografický protokol (bezpečnostní protokol nebo také **šifrovací protokol**) je protokol který zajišťuje bezpečný přenos informací na základě předem daných ustanovení. Protokol popisuje, jak by měl být daný algoritmus použit.

Protokol je v informatice **konvence nebo standard**, podle kterého probíhá elektronická komunikace a přenos dat mezi dvěma koncovými body (realizované nejčastěji počítači). V nejjednodušší podobě protokol definuje pravidla řídící syntaxi, sémantiku a synchronizaci vzájemné komunikace. Protokoly mohou být realizovány hardwarově, softwarově a nebo kombinací obou.

Kryptografický protokol se především používá pro bezpečný přenos dat na aplikační úrovni. Kvalitní kryptografický protokol obecně neumožňuje účastníkům získat data nebo provést jiné akce než protokol povoluje. Také získání dat běžným luštěním šifer musí být co nejsložitější a záznamů o původním textu musí být co nejméně, aby se omezilo prolomení šifry a získání původní zprávy. Jednou z nejpoužívanějších metod prolamování šifer je řešení hrubou silou.

Aby se předešlo prolomení klíče, používají kryptologické protokoly různé metody, funkce a jejich kombinace k tomu, aby své zprávy co nejlépe uchránily. Běžně kryptologické protokoly tedy podporují několik z těchto prvků:

- **Key-agreement protokol nebo ustanovení klíče**
- **Ověření entit**
- **Symetrické šifry**
- **Zabezpečený přenos dat na aplikační úrovni**
- **Metody nespornosti přenosu**

Široké spektrum kryptografických protokolů se nepoužívá jenom pro tradiční důvěryhodnost údajů, integritu a autentizaci, ale také pro řadu dalších požadovaných charakteristik zabezpečení v rámci počítačové spolupráce. Tzv. **slepé podpisy (Blind signatures)** se používají pro virtuální měny a elektronické pověření. Například pro ověření, že daná osoba má vlastnost nebo právo k danému úkonu, bez odhalení její identity či skupiny identit, se kterou osoba prováděla transakce. **Časová razítka** se používají pro ověření existence dat v určitém čase (a to i ověřených dat).

Bezpečný **multi-party výpočet** se používá pro výpočet odpovědí (jako jsou odhady největšího příhozu v aukcích) na základě důvěrných dat (jako jsou privátní příhozy) a to tak, že po dokončení výpočtu účastníci znají pouze své vlastní vstupy a výsledný odhad. Nepopíratelné podpisy jsou interaktivní protokoly, které umožňují osobě, která je podepsala, prokázat případné padělání a určit omezení, kdo může podpis ověřovat. Popiratelné podpisy rozšiřují standardní šifrování tím, že znemožňují případnému útočníkovi matematicky dokázat existenci otevřeného textu zprávy.

- Základní cíl kryptografie je řešit problémy, týkající se utajení, potvrzeni totožnosti, integrity zpráv a nečestných lidí.
- Protokol v tomto smyslu je série kroků, zahrnujících dvě a více stran, který má splnit úkol.
- Kryptografické protokoly jsou protokoly, které používají kryptografii (kryptografická primitiva).
- Základní bod používaní kryptografie v protokolech je v zamezení a detekci odposlouchávání a podvádění.

Kryptografické protokoly jsou protokoly, zajišťující bezpečný přenos informací na základě daných ustanovení. Protokoly popisují, jak by měl být daný algoritmus použit.

Charakteristiky kryptografických protokolů:

- Každý, kdo protokol používá, musí znát protokol a všechny jeho kroky v předstihu
- Každý, kro protokol používá, musí souhlasit s posloupností kroků
- Protokol musí být bez možností chyb
- Protokol musí být úplný

Typy protokolů:

- **Potvrzované protokoly** – protokoly, kde komunikaci potvrzuje nezávislá důvěryhodná osoba
- **Posuzované protokoly** – protokoly, u nichž je možné v případě sporu stran najít viníka
- **Samoopravné protokoly** – protokoly, kde komunikují pouze zúčastněné strany a součástí protokolu je identifikace viníka v případě porušení protokolu

Příklady:

Výměna klíčů, Autentizace, Autentizace s výměnou klíčů, Utajené rozdělení, Časové značky, Důkazy bez znalosti

předmětu, další protokoly ...

Šifrování

Šifrování je převod nezabezpečených (elektronických) dat za pomoci kryptografie na data šifrovaná a čitelná pouze pro majitele dešifrovacího klíče.

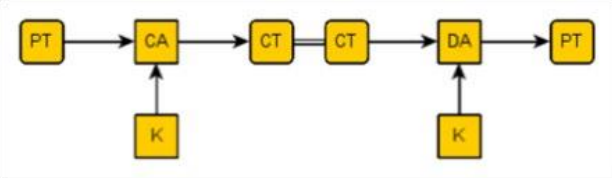
Symetrické šifry používají k šifrování i dešifrování stejný klíč. Výhodou je jejich nízká výpočetní náročnost. Ovšem nevýhodou je nutnost sdílení tajného klíče. Což znamená, že odesílatel a příjemce tajné zprávy se musí předtím domluvit na tajném klíči. Často se používají v kombinaci s asymetrickými šiframi, kdy se otevřený text zašifruje symetrickou šifrou a náhodně vygenerovaným klíčem. Tento symetrický klíč se zašifruje veřejným klíčem asymetrické šifry, takže dešifrovat data může pouze majitel tajného klíče dané asymetrické šifry.

Historie: Kryptografie do 20. století = šifrování (= převod informace z čitelního stavu do nečitelního). V 20. století, během WWII se objevily nové metody šifrování a dešifrování, založené na použití specialných mechnismů (např. Enigma – přenosný šifrovácí stroj, používaný k šifrování a dešifrování tajných údajů během WWII). V dnění době krytografie je blízko spojená s matematickou teorií a computer science, kryptografické algoritmy jsou navržené tak, aby jejich výpočetní složitost byla dostatečně velká.

Šifrování s tajným klíčem (symetrické šifrování)

1. blokové šifry: jednotka operace je blok (64bitů, 128 bitů), různé varianty vzájemného ovlivnění bloků (tzv. mody), každý blok se samostatně zpracován
2. proudové šifry: jednotka operace je bit či znak, operace s bitem může záviset na předchozích bitech

Symetrická kryptografie: Prostý text se s pomocí šifrovacího algoritmu převeden šifrovaný text a ten s pomoci dešifrovacího algoritmu zpět na prostý text. Výsledek šifrovacího algoritmu záleží na klíči



Vernamova šifra (jednorázová tabulková šifra) je jednoduchý šifrovací postup patentovaný v roce 1917 Gilbertem Vernamem. Spočívá v posunu každého znaku zprávy o náhodně zvolený počet míst v abecedě. To se prakticky rovná náhradě zcela novým písmem a na tomto faktu je založen důkaz, že Vernamova šifra je prakticky nerozluštitelná.

Postup šifrování probíhá tím způsobem, že vezmeme jednotlivá písmena tajné zprávy a každé posuneme o několik pozic v abecedě. Posloupnost čísel posunutí je následně také klíčem k rozluštění oné šifry. Takže klíč je vlastně stejně dlouhý, jako přenášená zpráva, což je také jednou z podmínek spolehlivosti. Další dvě podmínky spolehlivosti zní, že klíč musí být dokonale náhodný a každý klíč se může použít pouze jednou.

Symetrické blokové šifry, inicializační vektor

Symetrická bloková šifra je šifra, která pracuje s bloky pevně stanovené délky (nejčastěji 64 bitů (AES 128 bitů)). Delší zpráva se rozdělí do více bloků a ten poslední se následně doplní o nějakou výplň. Každý blok je následně zašifrován pomocí šifrovacího algoritmu, řízeného utajeným šifrovacím klíčem, poté je každý blok přenesen a následně rozšifrován stejným šifrovacím algoritmem pomocí stejného šifrovacího klíče.

Slabinou je opakované použití stejného klíče na všechny bloky. Řešení tohoto problému je použití Inicializačního vektoru. To je blok náhodných dat, který zajistí, aby výsledkem šifrování dvou identických zpráv byly dvě různé šifrované zprávy. Konkrétní příklady – DES, AES, IDEA, Triple DES, RC2, RC5.

Bezpečnostní slabiny ECB (Electronic Codebook) módu a jejich řešení

Název „codebook“ od toho, že je (teoreticky) možné mít tabulku kde ke každému bloku běžného textu je přiřazen blok šifrovaného textu. Pro 64 bitové bloky má tabulka šifer 264 položek.

Výhody ECB:

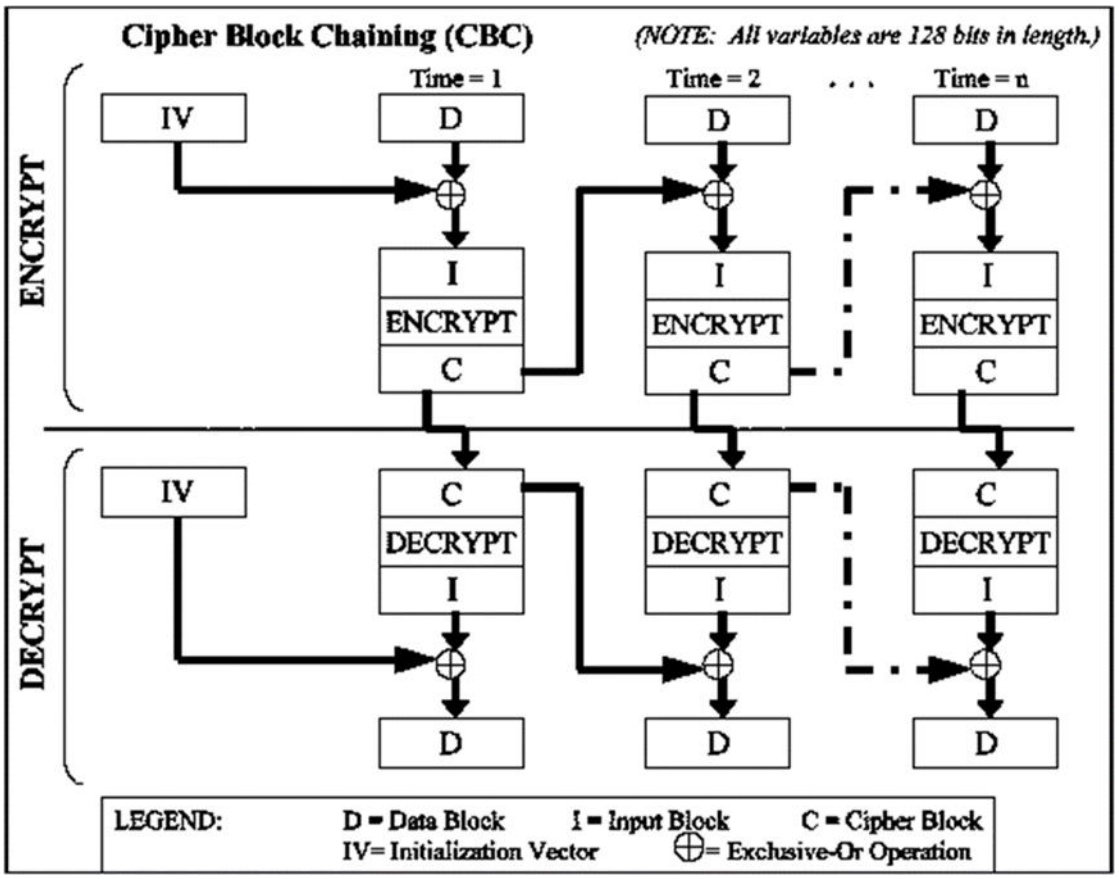
- Každý blok je šifrován nezávisle.
- Jednotlivé bloky je možné šifrovat v libovolném pořadí.
- Vhodné pro přístup k databázím.
- Možná paralelizace.

Nevýhody ECB:

- Tabulku šifer je možné začít konstruovat i bez znalosti klíče. Ve většině situací mají části zprávy tendenci se opakovat. Začátky zpráv obsahují hlavičky, které mohou být velice podobné, podobně i konce zpráv. Zprávy mohou mít parametry které mohou nabývat pouze několika hodnot.
- Vážnější problém je, že útočník může modifikovat obsah zprávy bez toho, aby byla modifikace zachycena na přijímací straně bez znalosti klíče. (dokonce bez znalosti algoritmu).

Šifra ECB (Electronic codebook) šifruje data každého bloku odpovídajícím algoritmem. Identické bloky budou proto vypadat stejně i po zašifrování, stejné bloky otevřeného textu mají tedy vždy stejný obraz. To dovoluje útočníkovi bloky šifrovaného textu libovolně vkládat, zaměňovat nebo odstraňovat.

Řešením této slabiny by mohlo být například použití módu CBC (Cipher Block Chaining Mode). Ten funguje na principu řetězení šifrovaných bloků, každý znak otevřeného textu ovlivňuje pouze jeden znak šifrovaného textu, každý další blok je šifrován pomocí předcházejícího bloku a to tedy způsobuje fakt, že identické nezašifrované bloky nejsou po zašifrování stejné. Nejprve je však důležité proměnit první blok náhodným inicializačním vektorem, který se následně pošle otevřeně přijímací straně, před šifrovým textem a až poté se zašifruje.



Informace o CBC z přednášek:

- CBC používá zpětnou vazbu, při níž je výsledek šifrování předchozího bloku zařazen do šifrování současného bloku.
- U CBC je volný text nejprve XORován se zašifrovaným textem předchozího bloku a teprve pak je zašifrován pro výstup. Zašifrovaný text je jednak odeslán a jednak zaznamenán pro použití na dalším bloku.
- Dešifrování je stejně tak přímočaré. Zašifrovaný text je jednak dešifrován a jednak zaznamenán v registru. Poté co je následující blok dešifrován je provedeno na něm XOR s uloženým blokem.

Inicializační vektor v CBC:

- CBC přiměje identický blok k zašifrování na jiné hodnoty pouze v případě, že část předchozího textu byla různá.
- Dvě stejné zprávy se zašifrují do stejného šifrovaného textu a ještě více různé zprávy, které stejně začínají se zašifrují do stejného šifrovaného textu až do prvního rozdílu. Způsob jak tohle potlačit je poslat v prvním bloku náhodná data nazývaná inicializační vektor (IV).
- IV není třeba držet v tajnosti ale měl by být změněn s každou zprávou

Vytvoření jiných kryptografických primitiv z blokové šifry (autentizace zpráv, proudová šifra a další).

Symetrické šifry jsou často používány k dosažení jiných kryptografických primitiv než jen šifrování. Šifrování zprávy nezaručí, že se tato zpráva nezmění. Proto se často autentizační kód zprávy přidá do šifrovaného textu, aby se tak

zajistilo, že změny v šifrovaném textu budou přijímačem rozpoznatelné. Také ověřovací kódy zpráv mohou být vytvořeny ze symetrických šifer (např. CBC-MAC). Symetrické šifry nemohou být použity k jiným účelům než k ověřování originality dat..

Proudové šifry a jejich výhody. V současnosti doporučované šifry a délky klíčů.

Šifry zpracovávají text po jednotlivých bitech. Vstupní datový tok je kombinován (typicky pomocí funkce XOR) s pseudonáhodným proudem bitů vytvořeným z šifrovacího klíče a šifrovacího algoritmu. Výsledkem je zašifrovaný datový tok, který je šifrován neustále se měnící transformací (na rozdíl od blokové šifry, kde je transformace konstantní). Proudové šifry jsou typicky rychlejší než blokové šifry a pro implementaci potřebují jednodušší hardware. Naopak jsou na rozdíl od blokových šifer náchylnější ke kryptoanalytickým útokům, pokud jsou nevhodně implementovány (počáteční stav nesmí být použit dvakrát). Konkrétní případy – RC4, FISH.

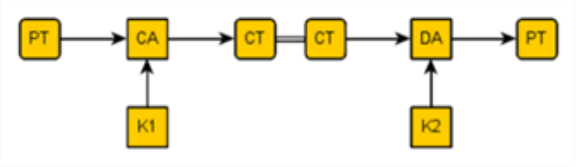
V současnosti používané symetrické šifry

V současné době je používána například symetrická šifra AES, kterou od roku 2002 používá jako federální standard USA se svou velikostí bloku 128 bitů a velikostí klíče 128, 192 nebo 256 bitů.

Asymetrická kryptografie

Principy asymetrické kryptografie; její využití pro výměnu klíčů, digitální podpisy a šifrování.

Šifrování s veřejným klíčem: Algoritmus má dva klíče svázané matematickým vztahem. Pomocí prvního se text zašifruje a pomocí druhého dešifruje. Jeden z klíčů je veřejný, druhý soukromý.

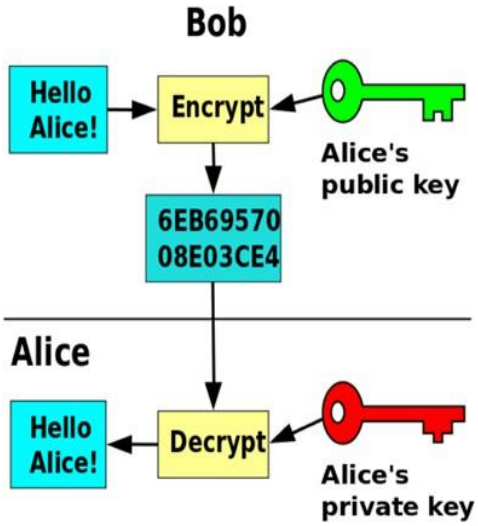


Asymetrická kryptografie je založena na tzv. jednocestných funkcích a obtížně řešitelných matematických problémech, což jsou operace, které lze snadno provést pouze v jednom směru, tzn., že ze vstupu lze snadno spočítat výstup, ale z výstupu je velmi obtížné nalézt vstup. Např. násobení: je velmi snadné vynásobit dvě velmi velká čísla, avšak rozklad součinu na činitele (faktorizace) je velmi obtížný. Asymetrické šifrování je podstatně náročnější než symetrické. Používá se také např. pro elektronické podpisy

Obrázky z wiki:

Vygenerování klíčě

Šifrování a dešifrování pomocí asymetrické šifry



Matematické základy:

Refaktorizace velkých čísel – RSA (Rivest Shamir Adleman 1978)

Faktorizace je problém rozložení čísla na součin menších čísel. V nejčastější podobě se používá jako rozklad celého čísla na součin prvočísel. Např. číslo 15 lze napsat jako 3*5. Rozklad celého čísla na prvočinitele je považován za velmi těžkou úlohu a na její nezvládnutelnosti pro velká čísla jsou založeny kryptografické metody jako např. RSA pro šifrování s veřejným klíčem.

Prolomení šifry (získání prostého textu z veřejného klíče a zašifrovaného textu) je ekvivalentní problému rozkladu součinu na dva činitele

Generování klíčů

- Vypočte se součin dvou velkých náhodných prvočísel p a q (pro nejvyšší bezpečnost by jejich délka měla být blízká)
    - -n = p ×q
- Vypočte se hodnota Eulerovy funkce $\varphi\varphi(nn) = (pp - 1)(qq - 1)$
- Zvolí se celé číslo e menší než $\varphi\varphi(nn)$, které je s $\varphi\varphi(nn)$ nesoudělné
- Najde se číslo d takové, aby platilo $dddd \equiv 1$ ($mmmmmm\ \varphi\varphi(nn)$), kde ≡ značí kongurenci zbytkových tříd
    - Použije se rozšířený Euklidův algoritmus na e a $\varphi\varphi(nn)$, čímž je nalezeno d a c do rovnice $dddd + cccc(nn) = 1$
- Pokud je e prvočíslo, tak $dd = (1 + rr \times \varphi\varphi(nn))/ee$, kde $rr = [(ee - 1)\varphi\varphi(nn)(ee-2)]$
    - Veřejným klíčem je dvojice (n, e), kde n se označuje jako modul a e jako šifrovací nebo veřejný exponent
    - Soukromým klíčem je dvojice (n, d), kde d se označuje jako dešifrovací nebo soukromý exponent

Šifrování

- Zpráva M je rozdělena do bloků menších než číslo n (pokud p a q byly asi 100ciferné, n bude asi 200ciferné a zpráva by měla být o něco kratší)
- Zpráva se bude skládat ze sady bloků přibližně stejné délky ci ci =Mie (modn)

Dešifrování

- Mi =cid (modn)

Bezpečnost

- Založeno na problému faktorizace
- Získání čísel p a q prolomí šifru
- Jiný způsob je odhad hodnoty (p-1)(q-1), ale není to lehčí problém

Diskrétní logaritmus (Diffie Helma)

Je založen na umocňování čísel $(AB) = (AC)$ $|(AB)| = |(AC)|$

- Účastníci se veřejně domluví na použitém modulu m a základu z
- Každý z účastníků umocní základ modulárně na svůj exponent a výsledek pošle dalšímu účastíkovi
- Algoritmus končí, když je každý z původních základů zpracován každým účastníkem

Nechť p, g, k, Y jsou přirozená čísla, pro něž platí , potom každé číslo k odpovídající uvedené rovnici nazveme diskrétním logaritmem o základu g z Y vzhledem k modulu p. Zatímco spočítat Y ze znalosti k, p, g je snadné, spočítat logaritmus k ze znalosti Y, p, g je velmi obtížné. To předurčuje tento problém k využití v asymetrické kryptografii.

Diskrétní logaritmus nad eliptickými křivkami

Užití:

- šifrování zpráv,
- domlouvání klíčů (key exchange),
- digitální podpis.

Asymetrická kryptografie – využití pro výměnu klíčů

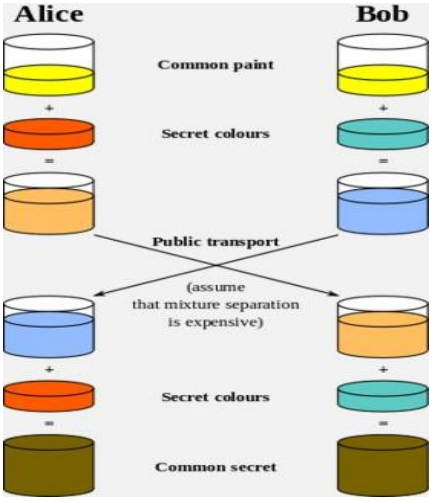
Asymetrický algoritmus používá dva klíče – jeden pro šifrování a jeden pro dešifrování. Pomocí klíče pro šifrování, který je veřejný a má k němu každý přístup je následně zašifrována zpráva, kterou může dešifrovat pouze majitel příslušného privátního klíče pro dešifrování.

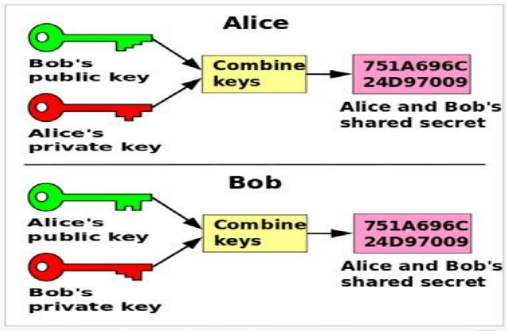
Využití pro výměnu klíčů je následující: např. server, který bude komunikovat s klientem použije jeho veřejný klíč k zašifrování klíče pro komunikaci, tento zašifrovaný klíč mu následně pošle. Nikdo kromě klienta tento klíč nemůže rozšifrovat, následně ho klient pomocí svého privátního klíče rozšifruje a oba mohou tento klíč používat bez obav, že by ho mohl sledovat i někdo jiný.

Key exchange (also known as "key establishment") is any method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm.

Příklad:

DH – Diffie Hellman, 1976 (ECDH – eliptické křívky Diffie-Hellman)





In the Diffie–Hellman key exchange scheme, each party generates a public/private key pair and distributes the public key. After obtaining an authentic copy of each other's public keys, Alice and Bob can compute a shared secret offline. The shared secret can be used, for instance, as the key for a symmetric cipher.

33. Certifikáty, elektronické podpisy, eIDAS (podepisování a identifikace)

Certifikát v IT

- Digitálně podepsaný veřejný šifrovací klíč vydaný certifikační autoritou
- Uchovává se ve formátu x.509
- Slouží k identifikaci strany v komunikaci
- Využívá princip přednostu důvěry

Obsah certifikátu

- Pořadové číslo certifikátu (SN) -unikátní číslo certifikátu
- Algoritmus podpisu pro výpočet otisku (např. SHA-1) a pro zašifrování otisku (např. RSA)
- Platnost (od kdy do kdy)
    - Po vypršení platnosti není možné certifikát používat např. k podepisování nebo šifrování, ale je nutné ho uchovat pro ověření nebo dešifrování zpráv s ním dříve podepsaných nebo zašifrovaných
- Vydavatel (Issuer) - Certifikační autorita, která certifikát vydala
- Předmět (Subject)
    - Specifikace držitele certifikátu
    - Pro jeden předmět by měl být vydán pouze jeden certifikát (nebo jeho prodloužení nebo obnovení po revokaci)
- Jedinečné jméno (Distinguished Name)
    - Skládá se z jednotlivých částí, jako je stát, město, organizace a další
- Veřejný klíč (Subject Public Key) o Veřejný klíč certifikátu, informace o algoritmu
- Rozšíření certifikátu (Extensions)
    - Další informace, které mohou být specifické pro určité programy
    - Pokud některá položka obsahuje příznak critical, ale program jí nerozumí, musí certifikát odmítnout
- Digitální podpis
    - Otisk certifikátu podepsaný vydávající certifikační autoritou (pro ověření, že je certifikát pravý)

Funkce

- Digitální podpis (Digital Signature)
    - K podepisování dat (autentizace uživatelů, integrita dat)
- Neodvolatelnost (Non-repudiation)
    - K ověření pravosti o Je nutný k ověření digitálního podpisu
- Šifrování klíče (Key Encipherment)
    - K šifrování soukromých (v asymetrické kryptografii) či tajných klíčů (v symetrické kryptografii)
    - Použití například při zasílání šifrované elektronické pošty:
        - Je vygenerován náhodný tajný klíč pro symetrickou šifru 405
        - Tímto tajným klíčem je zpráva zašifrována a tento tajný klíč je zašifrován veřejným klíčem příjemce a v zašifrované podobě zaslán společně se zašifrovanou zprávou
        - Příjemce zašifrovaný tajný klíč dešifruje svým soukromým klíčem a získaným tajným klíčem dešifruje původní zprávu –
- Zakódování dat (Data Encipherment)
    - K šifrování dat veřejným klíčem certifikátu
    - Toto použití není obvyklé –
- Domlouvání klíčů (Key Agreement)
    - K výměně klíčů
    - Použití v souvislosti s algoritmem Diffie-Hellman a u výměny klíčů s využitím eliptických křivek
    - U RSA nemá smysl
- Podepisování certifikátu (Key Certificate Sign)

    - Veřejný klíč certifikátu slouží pro ověřování certifikátů, jde tedy o certifikát certifikační autority
- Podepisování CRL (CRL Sign)
    - Certificate Revocation List
    - Slouží pro podepsání seznamu revokovaných certifikátů certifikační autoritou (pro možnost ověření, že je tento seznam autentický)
    - Pouze šifrování (Encipher Only) a pouze dešifrování (Decipher Only) o Použití s domlouváním klíčů (Key Agreement) o Dohodnutý klíč lze použít pouze pro šifrování, nebo pouze pro dešifrování

Digitální podpisy a generování klíčů

- Funkcí zprávy a utajené informace odesílatele (privátní klíč certifikátu), což jednoznačně označí odesílatele zprávy, a podpis je schopen verifikovat kdokoliv (ověření pravosti – původnosti)
- Zaručuje, že data nebyla cestou změněna (ověření integrity)
- Obsahuje podpisovou část a verifikační číst o Podpisová data mají utajovanou a veřejnou část
- V ČR je digitální podpis platný, pokud odpovídá zákonu číslo 227/2000Sb

Základní funkce

- Autentizace
    - Podpis nahrazuje ověření identity proti občanskému průkazu (spec. nároky na certifikát)
- Integrita
    - Změna zprávy po cestě je detekovatelná při doručení zprávy
- Nepopiratelnost
    - Odesilatel ručí za to, že má pod kontrolou svůj privátní klíč – kdo jiný by tedy zprávu poslal?
    - Součástí podpisu může být i časové razítko

Rizika

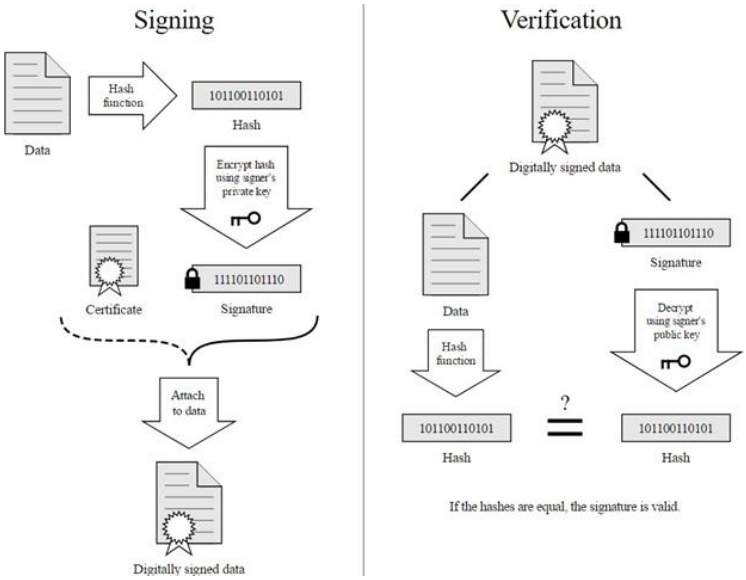
- Digitální podpis zprávy nešifruje
- Útočník může po cestě podpis ze zprávy odebrat a data zprávy změnit – doručí je pak bez podpisu jako normální zprávu a příjemce nic nepozná

Vytvoření podpisu

- Z dat je spočítán hash
- Hash je zašifrován privátním klíčem odesilatele a je k němu přiložen certifikát odesilatele
- Toto je spolu s daty odesláno

Ověření podpisu

- Zpráva se rozdělí na data a na podpis
- Z dat je vypočítán hash
- Je ověřena platnost certifikátu odesilatele doručeného v podpisu
- Podpis je dešifrován pomocí veřejného klíče odesilatele získaného z certifikátu -> hash
- Rovnají se oba hashe?



Algoritmy

- DSA – digital signature algorithm
- **Generování klíče**
    - p je prvočíslo L bitů dlouhé (L je mezi 512 a 1024 bity a je násobkem 64)
    - q je 160 bitový prvočíselný dělitel (p-1)
    - g = h (p-1)/qmod p, kde h je libovolné číslo menší než p-1 takové, že h (p-1)/qmod p je větší než 1
    - x je číslo menší než q a y = g x mod p

    - dále je použita hash funkce H(m)
    - parametry p, q a g jsou veřejné a mohou být společné pro skupinu uživatelů
    - x je privátní klíč a y je veřejný klíč
- **Vytvoření podpisu**
    - Alice generuje náhodné číslo k menší než q a dále:
        - r = (g k mod p) mod q
        - s = (k-1 (H(m) + xr)) mod q o parametry s a r jsou podpis a Alice je odesílá Bobovi
- **Verifikace podpisu**
    - Bob verifikuje podpis následujícími výpočty:
        - w = s -1 mod q
        - u1 = (H(m) × w) mod q
        - u2 = (rw) mod q
        - v = ((gu1 y u2) mod p) mod q
- Pokud se v rovná r, je podpis platný

Kvalifikované certifikáty, eIDAS

eIDAS – nařízení EU č. 910/2014 o elektronické identifikaci a důvěryhodných službách.

Pozor – je to vztah z hlediska komunikace “osoba – stát”.

Nařízení dohlíží na elektronickou identifikaci a důvěryhodné služby pro elektronické transakce na vnitřním trhu Evropské unie. Upravuje elektronické podpisy, elektronické transakce, definuje zúčastněné subjekty a procesy, aby zajistila bezpečnost pro uživatele podnikající on-line, například při elektronickém převodu finančních prostředků nebo při komunikaci s veřejnými službami. Nařízení eIDAS umožňuje bezpečně a pohodlně provádět transakce přes hranice bez užití tradičních papírových metod, jako jsou pošta nebo fax.

Nařízení vytvořilo standardy pro elektronické podpisy, kvalifikované digitální certifikáty, elektronické pečeti, časová razítka a další způsoby ověření autentizačních mechanismů. Ty umožňují, aby elektronická transakce měla stejné právní postavení jako transakce prováděná na papíře.

Nařízení vstoupilo v platnost v červenci 2014 jako prostředek k umožnění bezpečných a hladkých elektronických transakcí v rámci Evropské unie. Členské státy EU jsou povinny uznat elektronické podpisy, které splňují standardy eIDAS.

Typy elektronických podpisů:

- Zaručený elektronický podpis – z CESNETu, je uznáván všemi orgány veřejné moci v ČR.
- Uznávaný elektronický podpis – privátní klíč je uložen na disku. V ČR neexistují.
- Kvalifikovaný elektronický podpis – privátní klíč je uložen na nějakém prostředku, ze kterého jej nedostanete ani vy, ani útočník.

34. Identifikace a autentizace, hesla, vícefaktorová autentizace

Identifikace – Určité sdělení, kým jsem, za koho se vydávám, lze řešit například zadáním uživatelského jména, nebo například v různých souborech pomocí mého digitálního podpisu, nebo při přihlašování pomocí mého certifikátu.

Identifikace – tvrdím kdo jsem

Autentizace – Prokázání, že jsem skutečně tím, za koho se vydávám. V nejjednodušším případě to mohu prokázat pomocí hesla (prokazuji jeho znalost), pomocí jednorázového hesla nebo třeba autentizačního certifikátu

Autentizace – dokážu kdo jsem

- Prostředky k autentizaci
    - Informace, kterou vím – například právě již zmíněné heslo, to by však mělo být kvalitní (velká i malá písmena, číslice a interpunkční znaky a měla by být delší, také by je neměl používat nikdo jiný

- Předmět, který mám
  - Čipové karty
  - Generátory jednorázových hesel.
  - USB tokeny
- Něco co jsem
  - Otisky prstů a rukou
  - Parametry oka
  - Rozpoznávání obličeje

Vícefaktorová autentizace

Vícefaktorová autentizace je metoda ochrany přístupu pomocí kombinací již zmíněných prostředků k autentizaci. Většího zabezpečení můžeme dosáhnout použitím více prostředků a jejích větší kombinací.

Příklad:

Uživatel zná svoje přihlašovací jméno ve specifickém tvaru a heslo. Dále má na klíčích pověšený HW token, který musí následně doplnit ještě o PIN. – Máme dvě tajné informace, které musí uživatel znát a k tomu ještě jednu věc, kterou musí mít u sebe.

Jednorázová hesla

Jednorázové heslo (One time password) je heslo, které je platné pouze po dobu jednoho přihlášení nebo nějaké transakce. Tyto hesla se snaží vyhnout problémům, jako je například odposlechnutí hesla a znovupoužití. Pokud tedy útočník toto heslo zaznamená, k ničemu mu to nebude, protože už ho nemůže použít znovu.

Při vytváření těchto hesel se používají pseudonáhodné algoritmy. Je to kvůli tomu aby nebylo možné pozorováním starých hesel zjistit hesla nová. Hesla mohou být tedy generována:

- Na základě časové synchronizace mezi serverem a klientem
- Použitím matematického algoritmu, kde se nové heslo generuje na základě starého
- Použitím matematického algoritmu, kde je nové heslo generováno na základě nějaké činnosti (např. číslo zvolené autorizačním serverem, nebo detaily dané transakce)

Metody doručení jednorázových hesel jsou různé, dají se využít například mobilní telefony (SMS), osobní tokeny, nebo i generátory náhodných hesel, které mám při sobě.

Single-sign on

Způsob single sign on je způsob přihlášení založený na přihlášení na jedno použití, tento systém je dále dostupný v několika různých variantách přístupů.

- Například zde můžeme najít systémy založené na kerberosu. To je přístup, při kterém se uživatel přihlásí do systému a ten mu vygeneruje virtuální „vstupenku“. Ta je pak použita pro přístup k dalším aplikacím bez nutností přihlášení znovu. Tento přístup používá například windows u technologie aktivních adresářů.
- Smart-card-based - Initial sign-on prompts the user for the smart card. Additional software applications also use the smart card, without prompting the user to re-enter credentials. Smart card-based single sign-on can either use certificates or passwords stored on the smart card.

Musíme si však uvědomit že SSO přístup sebou nese i nějaká rizika. Největším problémem zde může být nekompatibilita aplikací třetích stran, kdy jejich vývojáři nedodržují normy a kompatibilitu rozhraní a problém tedy není na straně SSO.

Útoky na hesla

Kvůli stále častějším útokům na hesla je důležité, aby byla všechna hesla správně uložena a aby bylo použito správných funkcí pro zabránění útočníkům tato hesla lámat. Nejčastějšími chybami je například nepoužívání

správných zabezpečovacích funkcí (měly by se používat ještě neprolomené hashovací funkce a nejlépe ještě s přidáním soli při ukládání v databázích, tato sůl by neměla být stejná, jinak je pro útočníka mnohem jednodušší ji odhalit). Výrazně se nedoporučuje používat již prolomené hashovací funkce jako jsou například MD5 (byla prolomena, protože bylo zjištěno jak najít duplicitní slovo pro hash v rámci sekund).

Útočníci používají spousty různých způsobů, asi nejznámější z nich bude útok hrubou silou (brute force attack), kdy se hacker snaží uhádnout hesla bez jakékoliv znalosti dešifrovacího klíče. V praxi se jedná o systematické testování všech možných kombinací nebo omezené podmnožiny všech dostupných kombinací.

Dalším známým útokem je použití rainbow tables, což znamená, že si útočník stáhne do počítače spoustu již předem vypočtených hodnot s hesly, které následně porovnává s hodnotami ukradenými z databází. Na tomto způsobu je významně rychlejší to, že útočníkovi odpadá složité počítání hashů ale pouze je porovnává s hodnotami v tabulkách.

Kvalitní hesla:

- Používají velká i malá písmena
- Číslice a interpunkční znaky
- Dlouhá nejméně 8 znaků
- Místo hesel je vhodnější používat fráze.

Podpůrné prostředky:

- Kontroly proti slovníkům
- Kontroly znakové složitosti
- Povinný interval výměny
- Kontroly proti opakování

Autorizace/řízení přístupu určuje, zda může subjekt (uživatel, proces, atd.) provést požadovanou operaci (read, write, execute) na objektu (soubor, seznam, složka).

Modely přístupu

- Základní přístupové matice
    - UNIX, ACL
- Agregované přístupové matice
    - TE, RBAC, groups and attributes
- Plus Domain Transitions
    - DTE, SELinux, Java
- Formální přístupové modely
    - Bell-LaPadula, Biba, Denning
- Predikátové modely
    - ASL, OASIS, domain-specific models

DAC (Discretionary access model)

DAC neboli Discretionary access model je model, který umožňuje každému uživateli měnit stavy přístupu, pokud na to má dostatečná práva. Tento model je implementován ve standardních operačních systémech jako Unis, Linux, Windows

MAC (Mandatory access control)

MAC neboli Mandatory access control je model, který na rozdíl od modelu DAC neumožňuje uživateli měnit stavy přístupů, ale má je vynucena celosystémovými pravidly. Tyto systémy jsou postaveny na tom, že uživatelům se nedá věřit a říká se jim silně bezpečné systémy.

RBAC (Role based accesss control)

RBAC neboli Role based accesss control je model, který přiřazuje uživatelům práva k souborům dle jejich rolí (administrátor bude mít větší práva než běžný uživatel)

RBAC Products: SUN Solaris, Sybase SQL Server, BMC INCONTROL for Security Management, Systor Security Administration Manager

Bell-LaPadula model

Bell-LaPadula model definuje subjekty, objekty a přístupové operace mezi nimi. Dvě hlavní bezpečnostní pravidla tohoto modelu (jedno pro čtení a jedno pro psaní dat) jsou:

- subjekt nemůže číst data vyššího utajení, než je jeho oprávnění
- subjekt nemůže zapsat data nižšího oprávnění, než je jeho relace.

Biba model

Biba model se zaměřuje na integritu dat, a ne na důvěrnost.

No read down, no write up = Subjekt nemůže číst data o nižší důležitosti a nemůže zapsat data pro vyšší úroveň.

ACL (Access Control List).

ACL/Access Control List (seznam pro řízení přístupu) je vlastně seznam připojený k nějakému objektu (např. souboru), který určuje, kdo nebo co má povolení přistupovat k objektu a jaké operace s ním může provádět. V typickém ACL specifikuje každý záznam v seznamu uživatele a operaci. Např: Záznam v ACL [Pepa, smazat] pro soubor XYZ dává uživateli Pepa právo smazat soubor XYZ. U bezpečnostního modelu používajícího ACL tak systém před provedením každé operace prohledá ACL a nalezne v něm odpovídající záznam, podle kterého se rozhodne, zda operace smí být provedena.

35. Segmentace sítě a její provoz z pohledu bezpečnosti

Firewall

Slouží k řízení a zabezpečení síťového provozu mezi sítěmi s různou úrovní důvěryhodnosti a zabezpečení. Definuje pravidla pro komunikaci mezi sítěmi, které od sebe odděluje. Firewall je často používán společně s IDS.

Kategorie firewallů

Paketové filtry

- Filtruje pakety v závislosti na směru a portech (z jaké adresy a portu na jakou adresu a port může být paket doručen – kontrola na čtvrté vrstvě modelu OSI)
- Výhodou je vysoká rychlost zpracování
- Nevýhodou je nízká úroveň kontroly procházejících spojení – pro umožnění některých spojení (například streamování multimédií) je nutné otevřít množství portů a směrů, které potom mohou být využity i protokoly, které původně neměly být povoleny
- Nedokáže vyhodnotit obsah paketu
- Strategie dovolit vše/zakázat něco
- Strategie zakázat vše/dovolit něco

Aplikační proxy server (aplikační brány)

- Zprostředkovatel spojení (sítě jsou zcela odděleny)
- Klient (inicializátor spojení) ze sítě se obrací na proxy s žádostí o spojení, proxy na základě požadavku klienta otevře nové spojení k serveru, na kterém je klientem aplikační brána, a data jsou posílána přes tuto bránu
    - Kontrola se provádí na sedmé (aplikační) vrstvě OSI
- Pouze proxy smí přistoupit do sítě a pouze proxy je ze sítě vidět

    - Server nevidí zdrojovou adresu klienta, který je původcem požadavku (jako zdroj požadavku je uvedena vnější adresa aplikační brány) a aplikační brány tak fungují jako nástroje pro překlad adres (NAT)
- Výhodou je vysoké zabezpečení známých protokolů
- Nevýhodou je vysoká HW náročnost (aplikační brány zpracují výrazně nižší množství spojení a s menší rychlostí než paketové filtry a mají i vyšší latenci)
- Proxy musí být napsán pro každý protokol zvlášť
    - Lze použít generické proxy, které ale není bezpečnější než filtr paketů
- V Linuxu iptables
- Typy proxy serverů
    - Neupravuje odpověď – gateway nebo tunelové proxy
    - Forward proxy – internetově orientovaný
    - Reverzní proxy – internetově orientovaný

Kombinované firewally

- Filter – Proxy
- Filter – Proxy – Filter
- Vnitřní firewally v organizaci

Typy firewallů

Stavový:

- Paketový filtr s vědomím stavu TCP spojení (kontrola spojení na úrovni korektnosti procházejících dat známých protokolů i aplikací)
- pracuje na transportní vrstvě, rozlišuje stavy paketů na relaci a propouští pouze ty, které do patří do povolené relace, odporuje SPI (statefull packet inspection)

Bezstavový – rozhoduje se podle obsahu paketu

Techniky blokování:

- drop – paket se zahodí
- reject – pošle se ICMP (Internet Control Message Protocol) zpráva o nedostupnosti služby či protokolu
- tcp reset – pošle se TCP RST (při komunikaci přes protokol TCP)

Proxy servery

Alternativa k firewallům, ale fungují úplně odlišně. Proxy server funguje jako prostředník mezi klientem a cílovým počítačem (serverem), překládá klientské požadavky a vůči cílovému počítači vystupuje sám jako klient. Přijatou odpověď následně odesílá zpět na klienta. Může se jednat jak o specializovaný hardware, tak o software provozovaný na běžném počítači. Proxy server odděluje lokální počítačovou síť (intranet) od internetu.

Forward Proxy – dopředná proxy je představitelem klasického proxy serveru, umístěného blíže ke klientovi. Klient na dopřednou proxy zasílá dotaz s cílovou adresou a proxy vyřizuje komunikaci za něj. Příkladem je proxy server nastavený přímo v internetovém prohlížeči nebo caching proxy.

Reverse proxy – je proxy využívaná na straně serveru. Z pohledu klienta přistupujícího ke službě vystupuje zpětná proxy jako server, který může definovat obsah dostupný pro klienta. Zpětná proxy může zastupovat více serverů, mezi které pak dokáže rozložit zatížení. Slouží také k zprostředkování přístupu zvenčí k serverům, které jsou umístěny za firewallem v bezpečné vnitřní síti. Na straně klienta není třeba nic nastavovat.

NAT (Network Address Translation)

NAT se většinou používá v domácích sítích, kde máme private network. Chová se to velmi podobně jako proxy server. Server vidí pouze externí adresu.

NAT je proces, kdy dochází při průchodu NAT prvkem k výměně adresy v IP hlavičce za jinou. Primárním cílem je úspora IPv4 adres. NAT prvkem může být firewall, router nebo počítač. Na prvku, který provádí NAT, se vždy definuje jedno outside interface a jedno nebo více inside interface. Vnitřní síť pak často používá privátní adresy a vnější síť globální. Při překladu je vždy možné stanovit, které IP adresy se budou překládat a které ne pomocí access listů. Zabrání se tak zbytečnému překladu platných veřejných adres ve vnitřní síti.

IDS (Intrusion Detection System) a IPS (Intrusion Prevention System)

Pomocí databáze signatur a heuristické analýzy jsou schopny odhalit vzorce útoků i přede tím, než nastanou (např. skenování adresního rozsahu, rozsahu portů, …). Vyšší rychlost kontroly než při použití aplikační brány (proxy), ale pomalejší než použití paketových filtrů. Na rozdíl od firewallů tyto systémy řeší i útoky zevnitř sítě (firewall chrání síť od nebezpečných aktivit přicházejících z vně sítě).

- Centralizovaná správa
- Monitorování síťového provozu
- Skenování portů
- Logování a upozorňování (Alert)
    - Upozornění správce, provedení zápisu do logu, pokus o zastavení hrozby
- Typy
    - Host based (Uzlově orientované systémy detekce odhalení průniku)
        - Softwarový agent, který se snaží detekovat útoky pomocí analýzy systémových volání, činnost aplikací, úprav souborového systému a jiných akcí hostitele
- Network based (Síťově orientované systémy detekce odhalení průniku)
        - Nezávislá platforma, která identifikuje narušení tím, že zkoumá síťovou komunikaci a monitoruje více hostitelů
        - Senzory bývají umístěny přímo na síťových prvcích (huby a switche nakonfigurované pro zrcadlení portů) pro zachycení veškerého síťového provozu
        - Systém analyzuje všechny pakety, které procházejí sítí a snaží se v nich odhalit škodlivý kód
- Reakce
    - Pasivní
        - Při odhalení podezřelé aktivity nijak nezasahuje do síťového provozu
        - Pouze vygeneruje Alert a vytvoří záznam do logu
- Aktivní (IPS)
        - Proti podezřelé aktivitě kromě vygenerování Alertu a logu proti aktivitě zasáhne, například zablokováním služby
        - IPS při podezřelé aktivitě restartuje spojení nebo překonfiguruje firewall tak, aby blokoval provoz z podezřelého zdroje
- Detekce
    - Statistiky
    - Charakteristiky

Honeypot – informační systém, jehož účelem je přitahovat potenciální útočníky a zaznamenat jejich činnost. Honeypoty detekují činnost neoprávněných zdrojů přicházejících do systému. Tato detekce je po odhalení útočníka plně automatická. Automaticky se sbírají data o činnosti potenciálního útočníka. Detekce buď vyloučí, že se jednalo o útočníka, nebo to jen potvrdí. Je to rychlejší, než kdyby se sbírala data z funkčního napadeného systému. Honeypoty se někdy sdružují do sítě, tzv. honeynetu.

Web Application Firewall chrání webové stránky a webové aplikace před útočníky, kteří využívají zranitelná místa aplikace nebo protokolů ke krádeži dat nebo ke změně vzhledu webových stránek organizace.

Chrání před útoky na webové aplikace a útoky typu odepření služby (denial of service, DoS). Na rozdíl od tradičních síťových firewallů nebo detekčních systémů průniku (IDS), které jednoduše propouštějí HTTP, HTTPS nebo FTP provoz do webových aplikací, Web Application Firewall funguje jako obousměrný proxy tohoto provozu. Kontroluje, zda provoz neobsahuje útoky, izoluje webové servery od přímého přístupu hackerů. Kromě toho Web Application Firewall eliminuje útoky prováděné záměrnými změnami dotazů aplikací (např. znemožňuje úpravy cookies).

Na rozdíl od detekčních systémů průniku, které analyzují pouze binární vzorky, Web Application Firewall přejímá veškerý provoz místo webového serveru. Dekóduje komunikaci a odstraňuje/dropuje nepovolené znaky či dotazy a normalizuje data. Dále systémy umožňují ochranu proti zneužití citlivých údajů. Ze všech dnes hlášených útoků je zhruba 70 % cíleno na aplikační vrstvu.

Remote Access VPN

- VPN klient vytvoří další síťové rozhraní; je to na úrovni operačního systému
- Výhodou oproti SSH tunelu/proxy serveru je to, že na aplikacích není potřeba nic upravovat a nastavovat
- VPN server má v sobě NAT (Network Address Translation) – ta změní IP adresu odesílatele na IP adresu serveru
- Cílový server vidí spojení z VPN serveru

SSL (Secure Socket Layer)/TSL (Transport Security Layer)

SSL je předchůdcem TLS, oba protokoly jsou podobné

- Vrstva vložená mezi vrstvu transportní (např. TCP/IP) a komunikační (např. HTTP), která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran
- Typicky je autentizován pouze server (jeho totožnost je zaručena – například banka), ale klient zůstává neautentizován
- Při vzájemné autentizaci (serveru i klienta) je nutné použití infrastruktury veřejných klíčů (PKI)
- Nejčastější použití je pro zabezpečenou komunikaci s webovým serverem (protokol HTTPS)

Je možné použít i např. pro elektronickou poštu a další datové přenosy

Základní fáze fungování:

- Dohoda účastníků na podporovaných algoritmech
- Pro kryptografii s veřejným klíčem: RSA, Diffie-Hellman, DSA
- Pro symetrické šifrování: RC2, RC4, IDEA, DES, Triple DES, AES, Camellie
- Pro jednosměrné šifrování: Message-Digest algorithm (MD2, MD4, MD5), Secure Hash Algorithm (SHA-1, SHA-2)
- Výměna klíčů založená na šifrování s veřejným klíčem a autentizaci podle certifikátů
- Šifrování provozu symetrickou šifrou
- **Hand-shake (inicializace):**
    - Klient pošle zprávu ClientHello s nejvyšší verzí protokolu, kterou podporuje, náhodným číslem a seznamem doporučených šifrovacích sad a kompresních metod
    - Server odpoví zprávou ServerHello se zvolenou verzí protokolu, náhodným číslem a vybranou šifrovací a kompresní metodou
    - Server pošle svůj certifikát ve zprávě Certificate, pokud to zvolená šifra umožňuje
- Aktuálně používané certifikáty jsou typu X.509, ale existuje návrh na používání certifikátů vycházejících z OpenPGP
- Pokud má být autentizace vzájemná, může nyní server vyžadovat zprávou CertificateRequest certifikát od klienta
- Server pošle zprávu ServerHelloDone, čímž ukončuje inicializační proces dohody na používaných mechanismech
- Klient odpoví zprávou ClientKeyExchange, která může obsahovat tzv. PreMasterSecret, veřejný klíč nebo může být prázdná (podle zvolené šifry)
- Obě strany z náhodných čísel a hodnoty PreMasterSecret spočítají hodnotu MasterSecret, ze které jsou počítány veškeré další klíče pro komunikaci
- Klient odešle zprávu ChangeCipherSpect, která udává, že veškerá další komunikace bude šifrována
- Klient pošle zašifrovanou zprávu Finished obsahující hash a MAC (Message Authentication Code) předchozích zpráv

- Server zprávu rozšifruje a ověří hash a MAC, pokud dešifrování nebo ověření selže, měla by být komunikace ukončena
- Server pošle zprávu ChangeCipherSpec (další komunikace bude šifrovaná)
- Server pošle zašifrovanou zprávu Finished a ověření u klienta probíhá analogicky (viz. krok 9)
- Inicializace je dokončena a je povolen aplikační protokol, který bude šifrován

VPN (Virtual Private Network)

- Simuluje komunikaci počítačů v privátní síti přes veřejnou síť (přes internet)
- Každý účastník má virtuální MAC adresu (Media Access Control – jedinečný identifikátor síťového zařízení, fyzickou adresu) i IP adresu a komunikují, jako by byli v privátní síti
- Při navazování spojení je totožnost obou stran ověřena pomocí certifikátů, je provedena autentizace a veškerá komunikace je šifrována
- Použití například pro připojení k firemní síti odkudkoliv z internetu
    - Ve firemní síti musí být k dispozici VPN server připojený k internetu, ke kterému se klienti připojují, a který plní funkci brány zajišťující zabezpečení a šifrování komunikace
- Zobecněním VPN je síťové tunelování, kdy se na standartním síťovém připojení vytvoří virtuální linka mezi dvěma zařízeními, na které lze navázat další spojení

IPsec (IP security)

Bezpečnostní rozšíření IP protokolu založené na autentizaci a šifrování každého IP datagramu

- V architektuře OSI (referenční model ISO/OSI představuje řešení komunikace v sítích v jednotlivých nahraditelných vrstvách) se jedná o zabezpečení již na síťové vrstvě, a proto poskytuje bezpečnost jakémukoliv přenosu (jakékoliv síťové aplikaci)
    - Naopak bezpečnostní mechanismy vyšších vrstev nad protokoly TCP/UDP jako je SSL/TLS a SSH vyžadují podporu v síťové aplikaci
- Poskytuje ověřování dat, zajišťuje původnost dat, integritu a důvěrnost (šifrování)
- Princip:
    - Vytváří logické kanály – Security Associations (SA), které jsou jednosměrné a pro duplex se používají dva
    - Ověřování
        - Při přijetí paketu lze ověřit, zda odpovídá odesílateli a zda odesílatel existuje
    - Šifrování
        - Strany se dohodnou na formě šifrování paketu
        - IP paket je zašifrován buď bez hlavičky, nebo i s hlavičkou a je přidána nová hlavička
- Základní protokoly (většinou jsou používány zároveň)
    - Authentication Header (AH)
        - Zajišťuje autentizaci odesílatele a příjemce a integritu dat v hlavičce
        - Data nejsou šifrována
    - Encapsulating Security Payload (ESP)
        - Šifruje pakety
        - Vnější hlavička není chráněna ani není zaručena její integrita
- Provozní režimy
    - Režim přenosu
        - Je obvykle šifrován nebo pouze ověřován obsah paketu
        - Směrování paketu je nezměněno (pokud není upravena nebo šifrována hlavička paketu)
    - Režim tunelu
        - Je šifrován nebo ověřován celý paket včetně hlavičky a zapouzdřen do nového IP paketu s novou hlavičkou

Používán pro tvorbu VPN a pro tzv. host-to-network komunikaci (vzdálený přístup uživatele do sítě) a host-to-host komunikaci (například soukromý chat)