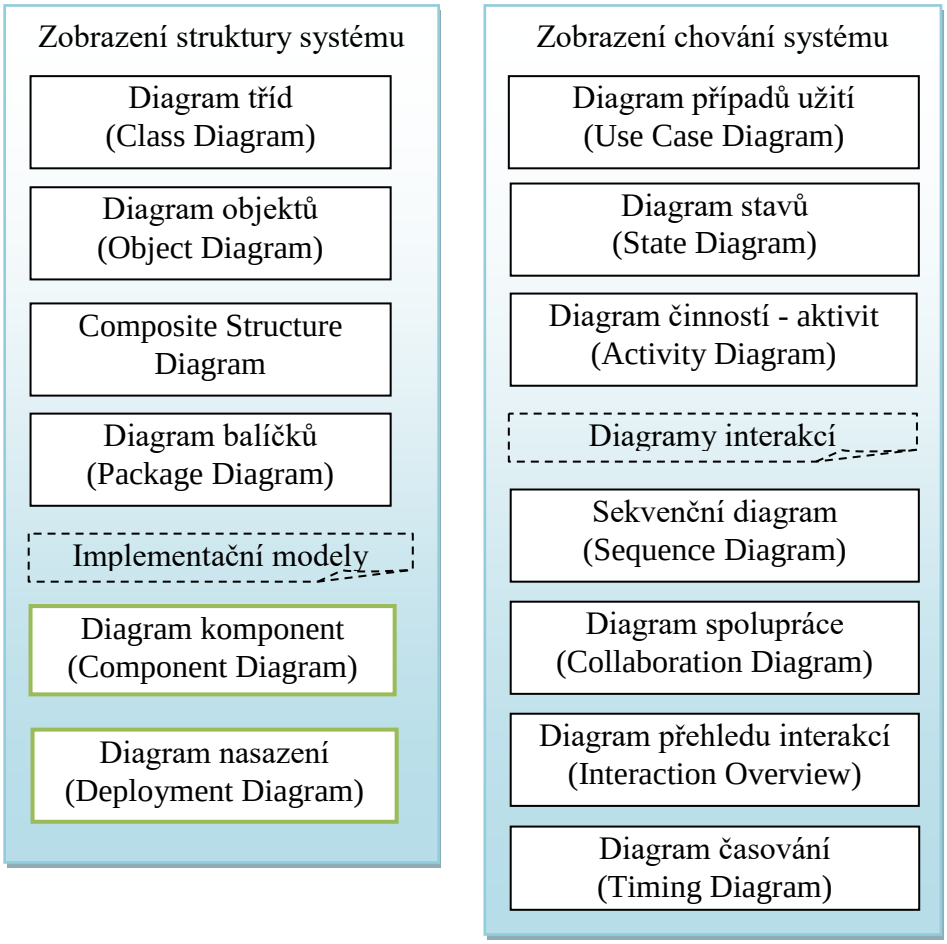
Předmět:	**Projektování IS**
Téma:	**Objektový přístup k vývoji IS (část E)**

Vyučující:	dr. Dušan Kajzar	Školní rok:	2020/2021

Obsah:

Kde se nacházíme ?



1. Diagram komponent (Component Diagram)

Účel modelu:

- znázornění softwarových komponent vyvíjeného IS,
- znázornění softwarové architektury systému (vazeb programových modulů).

Poznámka:

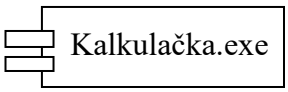
- srovnejme – model Structure Chart ve strukturovaném přístupu.

Které prvky (komponenty) zde zobrazujeme:

- spustitelné soubory (.exe, ...),
- dynamické knihovny,
- soubory s parametry,
- soubory s daty, …

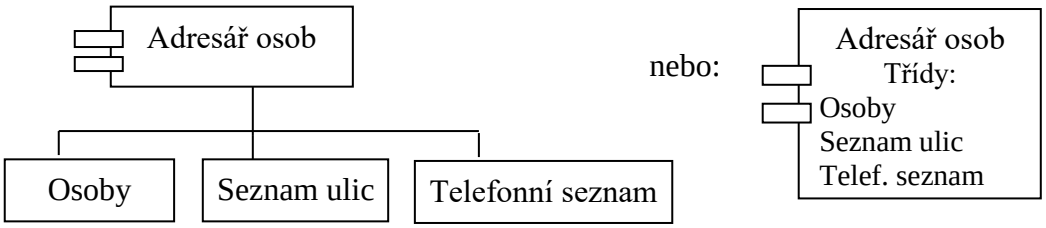
SW komponenta:

- obrázek - grafické znázornění SW komponenty.



SW komponenty vs. třídy:

- SW komponenta je softwarovou implementací jedné nebo více tříd,
- v diagramu komponent lze znázornit třídy, které daná komponenta implementuje,
- obrázek - třídy implementované komponentou "Adresář osob".

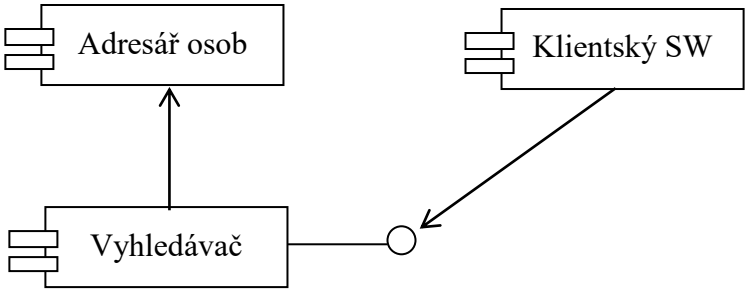


Rozhraní (interface):

- umožňuje opakované použití SW komponenty v různých subsystémech.

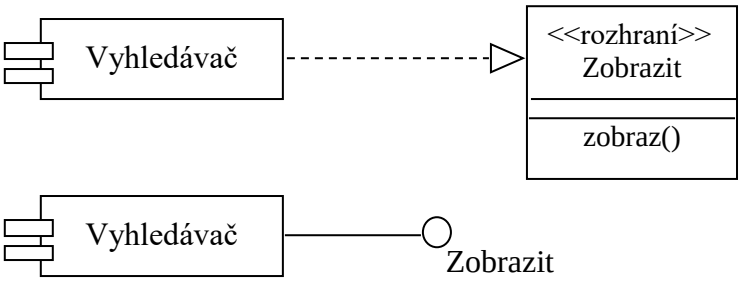
Příklad diagramu komponent:

- obrázek - interakce SW komponent v diagramu komponent.



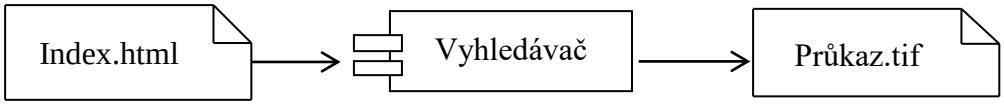
Znázornění rozhraní:

- obrázek - dva způsoby grafického znázornění rozhraní SW komponent.



Znázornění datových souborů:

- jde o soubory, se kterými SW komponenty pracují,
- využití stereotypu <<popis>>, nebo grafického znaku „Poznámka“.



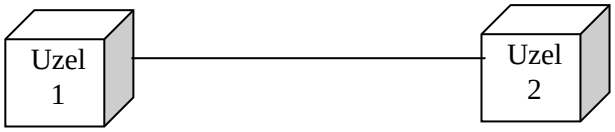
2. Diagram nasazení (Deployment Diagram)

Účel modelu:

- zobrazení hardwarové architektury systému,
- zobrazuje výpočetní (hardwarové) uzly a vazby mezi nimi.

Typy HW uzlů:

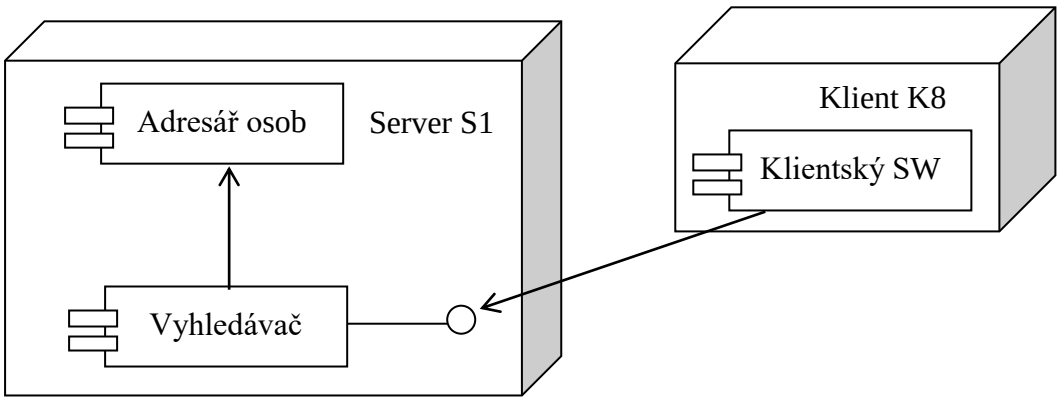
- aktivní uzel (procesor) – umí spouštět softwarovou komponentu,
- pasivní uzel (zařízení) – představuje rozhraní s vnějším světem (například tiskárna, monitor apod.),
- obrázek - HW uzly a jejich spojení.



Spojení mezi HW uzly:

- může znázorňovat různé druhy vazeb (obdobně jako u diagramu tříd),
- např. agregace, závislost apod.

Přiřazení SW komponent HW uzlům:



Dvoustupňový proces tvorby:

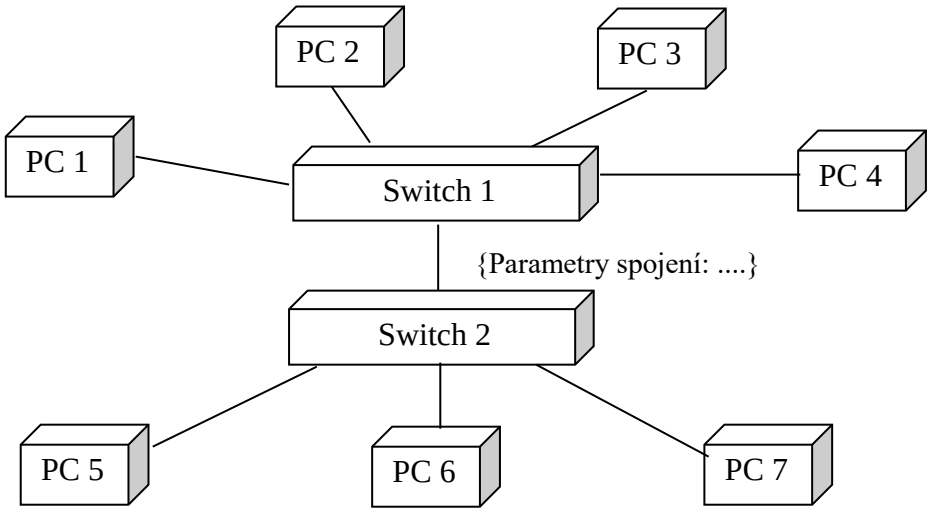
- 1. SW komponenty a vazby mezi nimi,
- 2. přiřazení SW komponent HW uzlům => uzly a jejich spojení.

Využití diagramu nasazení:

- modelování SW a HW architektury vyvíjeného systému,
- modelování počítačových sítí.

Zobrazení architektury počítačové sítě:

- obrázek - využití diagramu nasazení pro zobrazení architektury počítačové sítě.



Obrázky mimo jazyk UML:

- možnost použití jiných grafických symbolů - mimo jazyk UML,
- například obrázky počítačů (klientské stanice, servery),
- tiskárny, aktivní prvky sítě, obláček pro znázornění Internetu apod.,
- cíl - názornost zobrazení.

Poznámka:

- Setkali jsme se s obdobným diagramem v modelech strukturovaného přístupu?

3. Další diagramy v UML 2.5

Přehled:

- zdroj: http://www.uml-diagrams.org/uml-25-diagrams.html

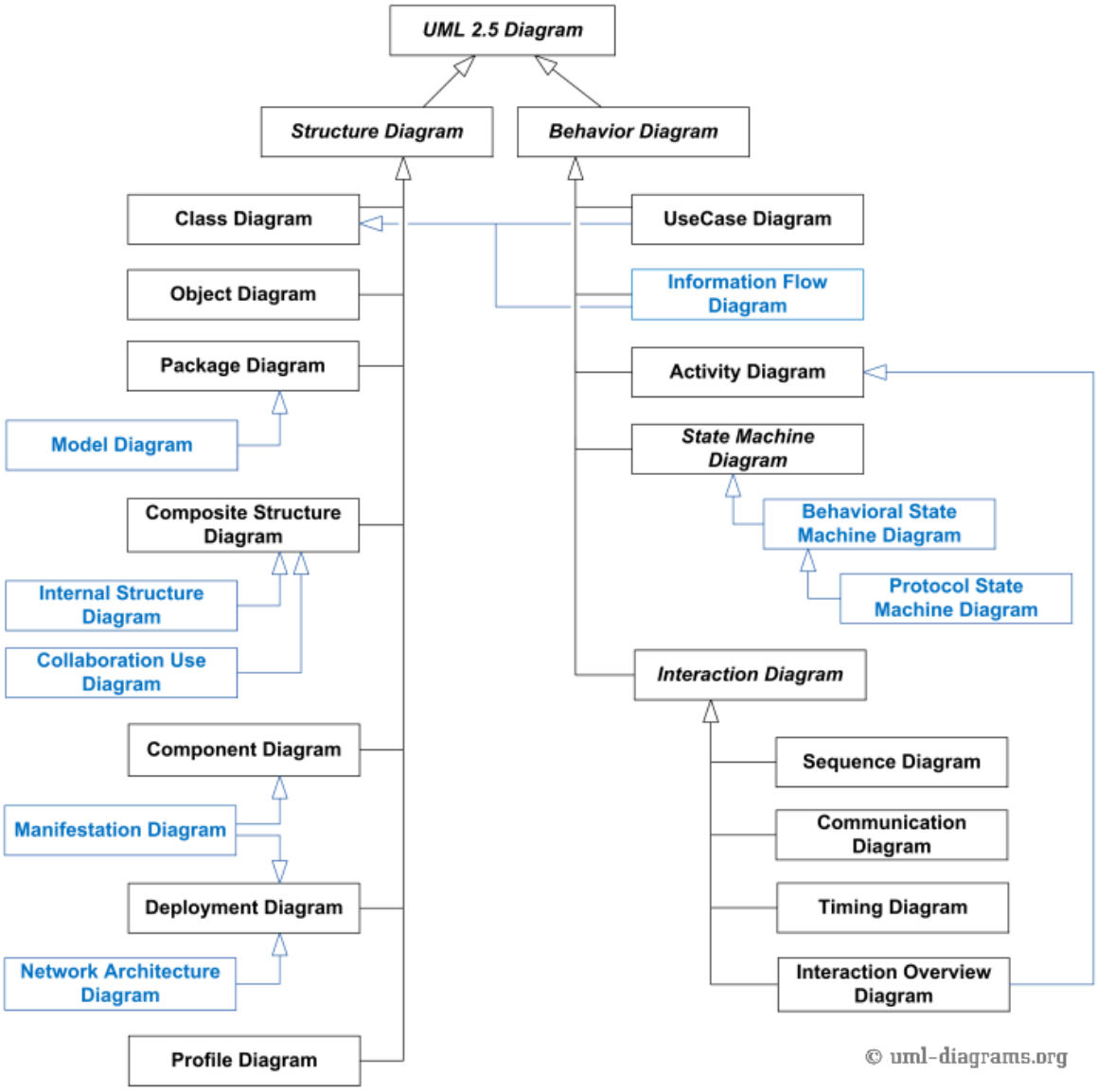


Diagram informačních toků (Information Flow Diagram):

- slouží ke zobrazení toků informací mezi prvky systému,
- mezi odděleními podniku, mezi subsystémy apod.

Příklad diagramu inf. toků:

- http://www.uml-diagrams.org/uml-25-diagrams.html

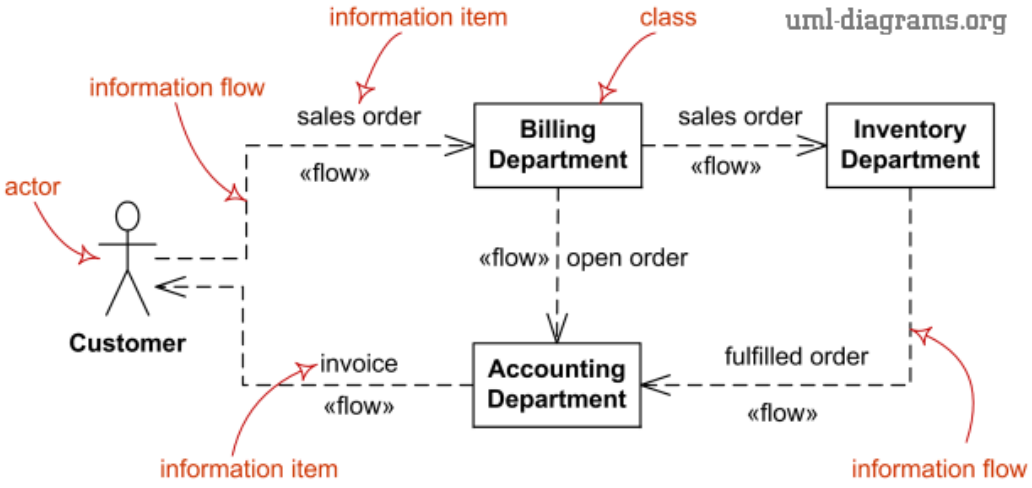


Diagram síťové architektury (Network Architecture Diagram):

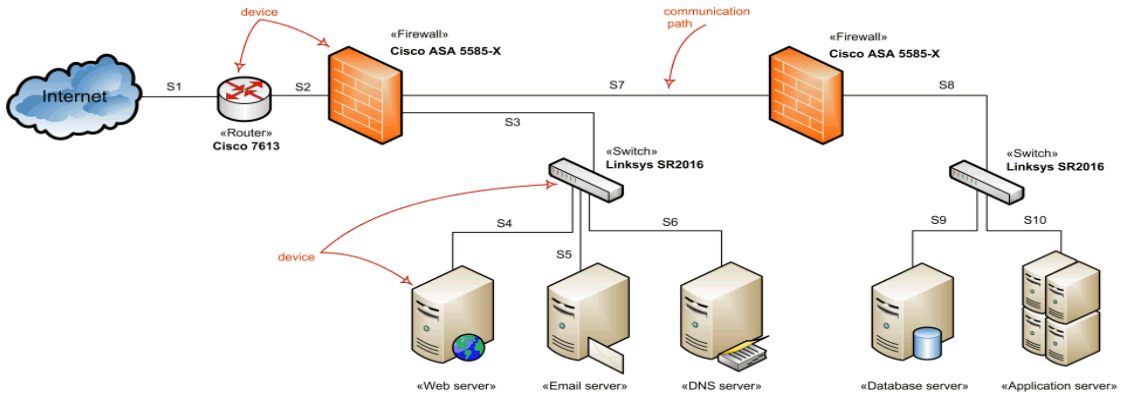
- slouží ke zobrazení počítačové sítě, její architektury,
- switche, routery, load balancery, firewally, ...

Poznámky:

- k tomuto účelu může být použit Deployment Diagram,
- ve verzi UML 2.5 hovoříme o spec. diagramu,
- používá grafické symboly mimo UML – obrázky.

Příklad Network Arch. diagramu:

- http://www.uml-diagrams.org/uml-25-diagrams.html



4. Tvorba modelů během vývoje IS

Kde se nacházíme?

- seznámili jsme se s modely jazyka UML,
- zkusme se nyní zamyslet nad posloupností jejich využití během vývoje IS,
- zatím bez vazby na konkrétní metodiku, pouze „selským rozumem“.

Mapování podnikových procesů (logika business procesů):

- použití modelů — Diagramy aktivit

Počáteční modely systému:

- Use Case diagram
- jakožto výchozí model znázorňující vyvíjený IS ve vztahu k jeho okolí,
- Diagram tříd
- znázorňující základní objekty (třídy) vyvíjeného IS a vztahy mezi nimi.

Rozpracování modelů:

- další rozpracování — Use Case diagram ⇨ Scénáře případů užití
- zdokonalujeme (zpodrobňujeme) — Diagramy tříd
- pro popisy činností využijeme — Diagramy aktivit

Spolupráce (interakce) mezi objekty během procesů:

- Diagramy sekvencí
- Diagramy spolupráce

} ⇨ doplňování informací do diagramu tříd

- Interaction Overview ... začlenění interakcí do kontextu aktivit

Přechod z analytické úrovně do úrovně návrhových modelů:

- doplnění návrhových tříd — Diagramy tříd
- potřebné zpodobnění — Diagramy sekvencí, Diagramy spolupráce

Úvahy nad životními cykly objektů (tj. instancí vybraných tříd):

- provádíme analýzu stavů objektů a změn stavů

  Diagramy stavů

- spolupráce objektů v různých stavech a v čase

  Diagramy časování

Návrh SW architektury:

- vytváříme Diagramy komponent

Kompletace SW a HW architektury systému:

- HW architektura systému Diagramy nasazení
- kompletace diagramů komponent a nasazení.

Další činnosti:

- návrh a schválení uživatelských rozhraní,
- návrh a schválení postupů testování systému,
- kompletace projektové dokumentace k systému,
- ...

5. Několik zásad pro modelování IS

Několik zásad modelování IS:

- zásada správnosti
  - dbát na úplnost, bezespornost, konzistenci modelů,
  - dbát na správnost syntaktickou (správné použití grafických znaků jazyka UML) a sémantickou (významovou),
- zásada důležitosti (závažnosti, relevance) a hospodárnosti (efektivnosti)
  - brát v úvahu informační potřebu cílového čtenáře,
  - model by neměl obsahovat více informací, než je nutné vzhledem k účelu jeho použití,
  - zvažovat poměr „náklady / využití / účel“,
- zásada srozumitelnosti
  - dbát na čitelnost a použitelnost modelů pro cílového čtenáře,

    - nikoliv čitelný pouze pro specialistu či pro autora,
- zásada srovnatelnosti
    - dodržovat stanovené konvence (dohody, úmluvy), standardy,
    - metamodely pro popisy obsahu modelů,
- zásada systematické struktury
    - modely = různé (doplňující se) úhly pohledu na danou realitu,
    - integrace pohledů => pochopení tvořeného systému.

Upozornění na některá nebezpečí:

- používání obecných vyjádření pro popisy činností
    - např.: „musí se shromáždit potřebné informace ...“
    - Jak shromažďujete potřebné informace? Kdo je shromažďuje? Kdy? ...
- vágní popis požadavků (nejasný, nejednoznačný, nepřesný)
    - např. „ovládání systému musí být intuitivní“,
    - „systém musí mít rychlé odezvy“, ...
- idealizace procesů a činností
    - znázornění, jak by to mělo být, nikoliv jak to skutečně je,
    - Skutečně to takhle děláte? Existuje situace, kdy postupujete jinak? Skutečně je to vždy první věc, kterou uděláte? apod.

6. Přehled historicky významných metod OOP

Poznámka k historii objektového přístupu:

- objektové programování,
- objektově orientovaná analýza a návrh IS.

Přehled významných O-O metod a technik:

- Booch (Object Oriented Design - OOD), publikována v roce 1991,
- Coad/Yourdon,
- OMT (Object Modeling Technique), James Rumbaugh, 1991,
- OMT2,
- OOSE (Objectory/Object-Oriented Software Engineering),
- Shlaer/Mellor,
- CRC (Class, Responsibility, Collaborators),

- BON (Business Object Notation), 1994,
- OOMT (Objektově orientované metodiky a technologie), VŠE Praha, 1996,
- BORM (Business Object Relation Modeling),
- RUP (Rational Unified Process), vyvinuta společností Rational Software Corporation,
- UP (Unified Process) - je otevřeným standardem procesu vývoje IS, vyvinuto autory jazyka UML.

Doporučená literatura:

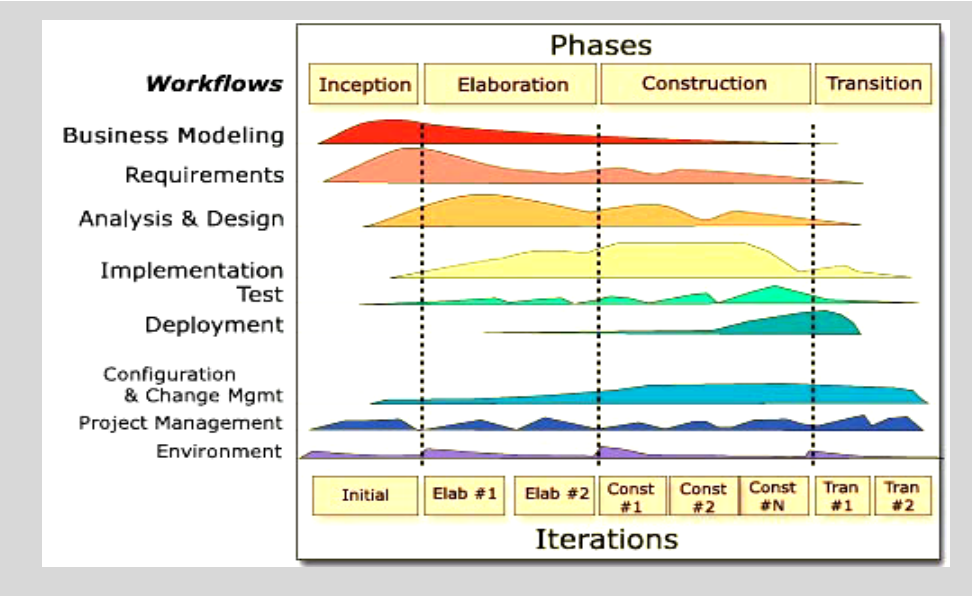
- Arlow J., Neustadt I.: „UML2 a unifikovaný proces vývoje aplikací“, Computer Press,
- Kadlec V: „Agilní programování“, Computer Press, 2004,
- Drbal P.: „Objektově orientované metodiky a technologie“, 1. a 2. díl. Vysoká škola ekonomická, Praha, 1997,
- Polák J., Merunka V., Carda A.: „Umění systémového návrhu“. Publishing, Praha, 2003.

7. Metodika RUP

RUP (Rational Unified Process):

- rozsáhlá propracovaná metodika,
- vyvinuta firmou Rational Software (byla koupena firmou IBM),
- komerční metodika, platí se za potřebný počet licencí.

Schéma metodiky RUP:



Vertikální osa - aktivity v každé iteraci:

- Business Modeling - modelování podnikových procesů,
- Requirements - práce s uživatelskými požadavky,
- Analysis & Design - analýza a návrh systému,
- Implementation - implementace systému,
- Test - testování, ověřování,
- Deployment - nasazení systému a jeho další rozvoj,
- Configuration & Change Management - řízení změn ve verzích systému a konfigurační řízení,
- Project Management - aktivity spojené s řízením projektu a řízením rizik,
- Environment - aktivity zaměřené na interní procesy a nástroje vývoje SW.

Horizontální osa:

- zahájení, rozpracování, konstrukce (realizace), zavedení.

Zahájení, inicializace (Inception):

- účel etapy
  - specifikace vize konečného produktu,
  - identifikace projektu, jeho cíle, účelu, koncepce, rozsahu,
  - identifikace uživatelských požadavků,
  - modelování podnikových procesů,
- hlavní výstupy etapy
  - dokumentace koncepce projektu, tj. základních požadavků klienta a vlastností produktu,
  - úvodní (základní) konceptuální model systému („hotový“ z 10% - 20%),
  - úvodní (základní) slovník projektu (jako doménový model),
  - základní podnikové procesy,
  - plán projektu,
  - základní analýza rizik projektu,
  - návrh jednoho resp. více prototypů.

Rozpracování (Elaboration):

- účel etapy
  - analýza a návrh systému,
  - rozbor a popis vlastností systému,
  - návrh kvalitní architektury systému,
  - plánování postupu projektu a potřebných zdrojů,
- hlavní výstupy etapy
  - konceptuální model systému (min. z 80% hotový),
  - identifikace všech uživatelů systému a služeb poskytovaných systémem,
  - úplný seznam uživatelských požadavků,
  - přesná architektura systému,
  - funkční prototyp systému,
  - podrobný další plán projektu,
  - revidovaná analýza rizik projektu.

Konstrukce, realizace (Construction):

- účel etapy
  - tvorba systému (programování),
  - inkrementální přidávání funkcionality do struktury systému,
- hlavní výstupy etapy
  - softwarový produkt na požadované platformě,
  - uživatelská dokumentace k systému,
  - popis aktuální verze produktu,
  - přesný popis architektury systému,
  - funkční prototyp systému,
  - podrobný další plán projektu,
  - revidovaná analýza rizik.

Zavedení, předání (Transition):

- účel etapy
  - dodání produktu koncovému zákazníkovi,
  - zaškolení uživatelů,
  - zavedení produktu do provozu,
  - podpora a další údržba produktu.

Výchozí principy pro práci v RUP (tzv. Best Practises):

- iterativní vývoj (Develop Iteratively)
  - vývoj softwaru postupným zjemňováním (refinement),
  - postupně pronikáme do řešené problematiky,
  - přidávání dalších vlastností (increments) v jednotlivých iteracích,
  - průběžně získáváme zpětnou vazbu ze strany uživatele,
  - postupným přibližováním se představám uživatele redukujeme rizika projektu,
- správa uživatelských požadavků (Manage Requirements)
  - tzv. přístup řízený případy užití (use-case-driven-approach),
  - pohled na výsledek každé činnosti jako na hodnotu pro uživatele,
  - neustálé doplňování, třídění a hodnocení požadavků zadavatele,

    - požadavky jsou dynamické – nutno počítat s jejich změnami,
    - celý vývoj IS je podřízen požadavkům zadavatele (uživatele),
- použití architektury komponent (Utilize Component Architectures)
    - myšlenka - bývá levnější koupit produkt splňující 75% požadavků, než vyvinout produkt, který nakonec splňuje také 75% požadavků,
    - aplikace vzorů, komponent, nákup a využití hotových modulů a částečných řešení,
- vizuální modelování (Model Visually)
    - postupné vytváření modelů - systému, procesů, ...
    - účel – dobré pochopení potřeb a požadavků pro analýzu, návrh, implementaci a další rozvoj systému,
- neustálé průběžné sledování a hodnocení kvality (Verify Quality)
    - kontrola kvality postupu vývoje IS,
    - kontrola kvality výsledného produktu,
    - snaha nalézt a odstranit problém co nejdříve,
- řízení a kontrola změn (Control Changes)
    - neřízené změny – zdroj chaosu, mohou rozvrátit vývojový proces,
    - proces řízení změn nutno integrovat do vývojového procesu.

Metodika RUP se také zabývá a popisuje:

- role (roles)
    - úlohy a pozice ve vývojovém týmu,
    - např. analytik, návrhář, implementátor, tester, manažer, recenzent, architekt atd.,
- meziprodukty (artifacts)
    - ve formě dokumentů, záznamů, požadavků, katalogů, diagramů,
    - např. model, prvek modelu (třída, subsystém,...), dokument, zdrojový kód, spustitelná aplikace,
    - jsou dodávané jako vstupy resp. výstupy jednotlivých aktivit,
    - vznikají a jsou modifikovány v průběhu procesu vývoje,
- aktivity (activities)
    - nejmenší jednotky práce (úkoly) v rámci procesu vývoje,
    - vykonávané jednou či více osobami v konkrétních rolích,

    - výsledkem je i vytvoření nebo modifikace příslušného dokumentu (např. zápis požadavků, identifikace tříd apod.),
- pracovní procesy (workflow)
    - sekvence navazujících aktivit, rolí a dokumentů,
    - časová posloupnost aktivit.

Vhodnost použití metodiky RUP:

- pro velké vývojové firmy (nikoliv pro jednoúčelové týmy),
- pro dlouhotrvající velké projekty,
- kde je nutné použít přísné, zdokumentované, propracované vývojové procesy,
- nutno vyčlenit čas na studium a implementaci metodiky RUP,
- adaptabilní – možno přizpůsobit i malým projektům a týmům.

Vztah RUP a UP (Unified Process):

- metodika UP je nekomerční, volně dostupná „varianta“ RUP,
- UP je jednodušší, méně propracovaná,
- UP – viz literatura „UML2 a unifikovaný proces vývoje aplikací“.