# 8 Přehled OO metodik (metod, metodologií)

## 8.1 OO metodiky konce 80. a začátku 90.let

- všechny populární OO metodiky předpokládají, že:
  a) zadavatel má jasný názor na svoje požadavky,
  b) zadavatel a vývojáři se dohodli na potřebě zdokumentovat požadavky ve formě nějakého dokumentu
- všechny populární OO metodiky předpokládají existenci počátečního zadání
- metodiky poskytují modelovací techniky a postup (+doporučení) pro vytvoření „dobrých“ modelů reprezentujících požadavky

- Rumbaugh – OMT (Object Modeling Technique)

  3 modely:
  - objektový – diagram tříd, diagram objektů
  - dynamický – diagram scénáře, stavový diagram
  - funkční – diagram datových toků (DFD)

**Fáze OMT:**

1. ***Analýza*** **- pochopení a modelování požadavků na vytvářenou aplikaci v kontextu aplikační domény.**

   **Vstup: zadání problému**
   **Výstup: formální model (specifikační dokument)**

2. ***Systémový návrh*** **- vymezení podsystémů, souběžných úloh**
3. ***Objektový návrh*** **- zpracování, zpřesnění, doplnění, optimalizace modelů analýzy, zahrnutí pojmů implementace**
   **- algoritmy,**
   **- optimalizace objektového modelu,**
   **- datové struktury (atributy, asociace),**
   **- uživatelské rozhraní,**
   **- moduly**
4. ***Implementace*** **- použití programovacích jazyků a relačních databází**
5. ***Testování***
6. ***Údržba***

**ad 1)**

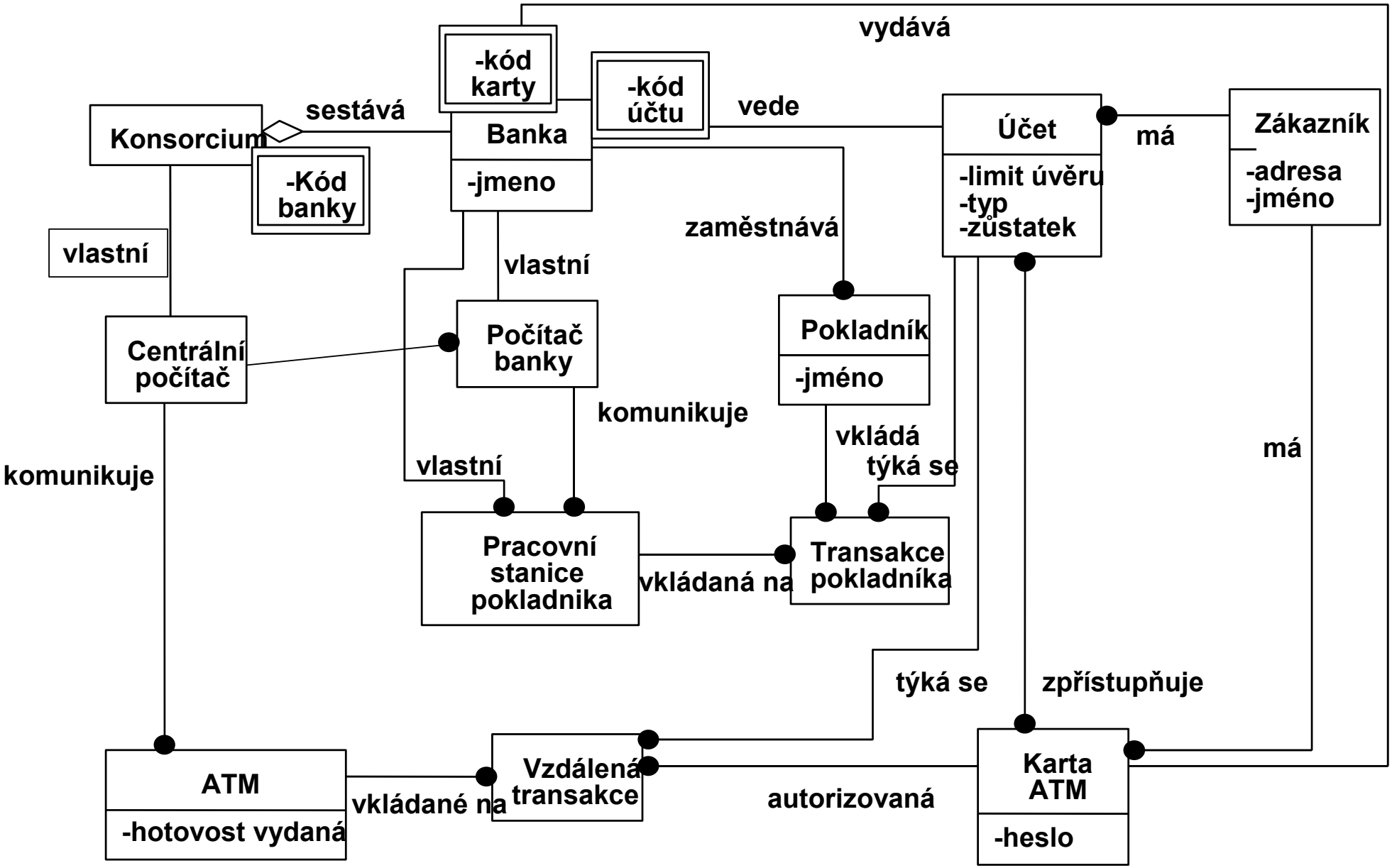
**Objektové modelování (diagram tříd aplikační domény):**

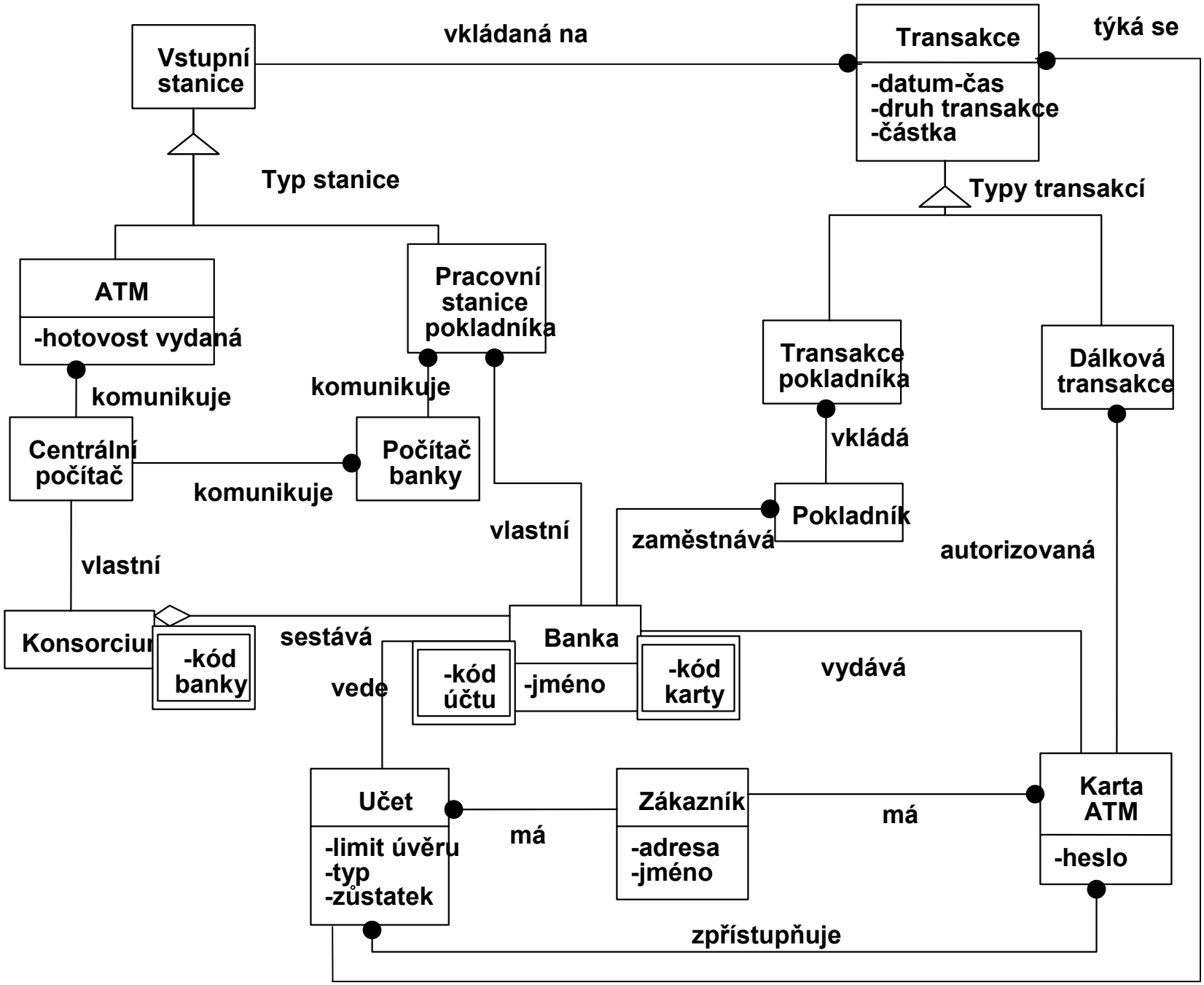
- **A) identifikuj třídy**
- **B) identifikuj asociace**
- **C) přidej atributy**
- **D) využij dědičnosti**
- **E) přidej operace**

} **využití slovního rozboru zadání**

**Dynamické modelování**

- **A) připrav scénáře typických posloupností interakcí**
- **B) identifikuj události mezi objekty**
- **C) připrav diagram scénáře pro každý scénář**
- **D) vytvoř stavový diagram**


vydává
-kód karty
-kód účtu
sestává
vede
Konsorcium
Banka
Účet
má
Zákazník
-Kód banky
-jmeno
-limit úvěru
-typ
-zůstatek
-adresa
-jméno
vlastní
zaměstnává
vlastní
Počítač banky
Pokladník
Centrální počítač
-jméno
komunikuje
vkládá
týká se
má
komunikuje
vlastní
Pracovní stanice pokladnika
vkládaná na
Transakce pokladníka
týká se
zpřístupňuje
ATM
-hotovost vydaná
vkládané na
Vzdálená transakce
autorizovaná
Karta ATM
-heslo


Vstupní stanice
vkládaná na
Transakce
týká se
-datum-čas
-druh transakce
-částka
Typ stanice
Typy transakcí
ATM
-hotovost vydaná
Pracovní stanice pokladníka
Transakce pokladníka
Dálková transakce
komunikuje
komunikuje
vkládá
Centrální počítač
Počítač banky
komunikuje
Pokladník
vlastní
zaměstnává
autorizovaná
vlastní
Konsorcium
-kód banky
sestává
Banka
-kód účtu
-jméno
-kód karty
vydává
vede
Učet
-limit úvěru
-typ
-zůstatek
Zákazník
-adresa
-jméno
Karta ATM
-heslo
má
má
zpřístupňuje

- **Další metodiky**
  - ➢ **Booch OOD**
    - **diagramy tříd**
    - **diagramy objektů**
    - **diagramy modulů (rozčlenění na moduly)**
    - **diagramy procesů (alokace procesů procesorům)**
    - **stavové diagramy**
    - **diagramy scénářů**

  **- síla v propracovaném návrhu**

  - ➢ **Jacobson OOSE**
    - **hlavním přínosem model použití (Use case),**
  - ➢ **Návrh řízený zodpovědností (Wirfs-Brock)**

  **- zodpovědnost tříd a spolupráce na základě metody CRC štítků**

  - ➢ **Coad/Yourdon, Martin/Odell OOIE, Shlaer-Mellor, FUSION (HP), Mainstream Objects (Software AG), …**

## 8.2 Metodika Rational Unified Process (RUP)

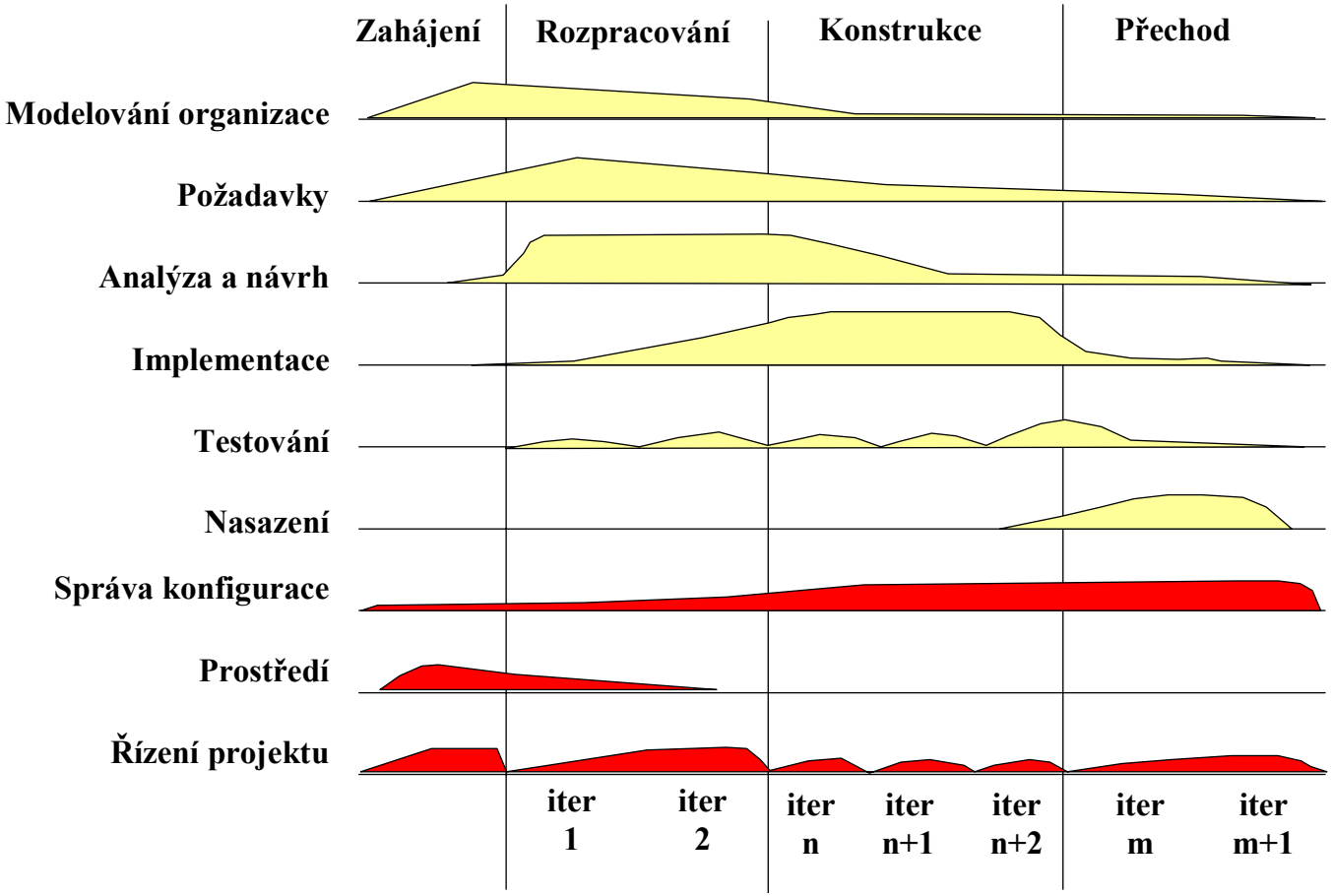
*Cíl:* **Produkce kvalitních programů, které splňují potřeby koncových uživatelů v rámci projektů s predikovatelným plánem a rozpočtem.**

*Vlastnosti:*

- **iterativní proces,**
- **důraz na tvorbu a údržbu modelů (přednost před papírovou dokumentací),**
- **důraz na architekturu systému (architecture-centric),**
- **řízený případy použití,**
- **objektově orientované techniky (UML),**
- **podporuje řízení kvality a správu rizik.**

*Fáze* **– interval mezi dvěma hlavními milníky procesu, kdy jsou splněny definované cíle, dokončeny produkty (artefakty) a učiněna rozhodnutí, zda přejít do další fáze.**

1. *Zahájení* **(inception) – podnikatelské zdůvodnění projektu.**
2. *Rozpracování* **(elaboration) – sestavení plánu a stanovení architektury.**
3. *Konstrukce* **(construction) – tvorba systému.**
4. *Přechod* **(transition) – dodání koncovým uživatelům.**




- každá fáze může zahrnovat řadu *iterací*.
- průchod všemi fázemi tvoří *vývojový cyklus*, výsledkem je jedna generace programového systému

**Př) proces používající kostry RUP (viz Fowler)**

***ad 1) Zahájení***

- různá podoba (interview .. několikaměsíční studie uskutečnitelnosti)
- vypracování podnikatelského záměru pro projekt
- nutná počáteční analýza pro představu o rozsahu

***ad 2) Rozpracování***

- cílem je lepší pochopení problému

● řízení *riziky*

a) *rizika požadavků* – uděláme něco jiného, než chce zákazník

- největší prostor pro použití UML
- nalezení všech, resp. nejrizikovějších *případů použití*
- vytvoření *konceptuálního modelu* domény (souběžně se sběrem případů použití):
  - ◊ *diagram tříd* pro konceptuální pohled
  - ◊ při výrazném toku aktivit *diagram aktivit* (paralelní procesy)
- používat minimální notaci, kreslení několika dílčích diagramů
- vytvoření jednoho konzistentního modelu domény s expertem na danou doménu, u velkých modelů rozčlenění na seskupení (moduly - package)

- je-li třeba, vytvoření *stavových diagramů* pro třídy se zajímavým životním cyklem
- vytvořený model je nutné chápat jako kostru pro zbytek modelu (je detailní, ale jen část) – co má tvořit kostru?
- malý tým pro tvorbu modelu domény (2-4 lidé: vývojáři+experti na doménu), problém doménové odbornosti
- využití prototypování pro rizikové případy použití

b) *technologická rizika* – vybíráme vhodné technologie?
- využití prototypování pro ověření technologií
- problémy při skloubení technologií a komponent návrhu
- zaměřit se na důležitá rozhodnutí návrhu architektury a oblasti, které se asi později dají těžko změnit
- techniky UML které lze použít: *diagramy tříd a interakce*, *diagramy seskupení, diagramy rozmístění*

c) *dovednostní rizika*– máme (můžeme získat) vývojáře s dostatečnou zkušeností?
- častý problém řady projektů
- často se vyplatí investovat do školení, začlenění konzultanta, resp. poradce do týmu (jinak plánovat průběžné oponování)

d) „*politická“ rizika* – existují „politické“ síly s vlivem na projekt?

- kdy je fáze ukončena?
  - trvání asi 1/5 doby projektu
  - indikátory:
    - vývojáři jsou schopni odhadnout pracnost každého případu použití
    - nejdůležitější rizika identifikována a pochopena

- plánování fáze konstrukce

  A) Kategorizace případů použití (UC) zákazníkem z hlediska „podnikatelské“ hodnoty (vysoká, střední, nízká)

  B) Kategorizace UC vývojáři z hlediska rizik

  C) Odhad pracnosti jednotlivých UC [člověkotýden (čt)] $EUC_i$

  D) Podívej se na nejrizikovější UC, je-li na ně třeba příliš času,je třeba jejich další rozpracování

  E) Určení doby iterace (několik UC, pro C++ např. 6-8 týdnů) *TI*

  F) Určení využitelné kapacity iterace a počtu iterací

  Př) iterace 6 týdnů, 8 vývojářů, korekce 2: $EI = (6*8)/2 = 24$ čt

  Počet iterací: $NI = \sum EUC_i/EI + 1$

  G) Přiřazení UC iteracím – nejprve UC s vysokou hodnotou a rizikem

  H) Pro fázi přechodu plánuj10-35% času konstrukce a ještě přidej 10-20% času konstrukce jako „faktor nahodilosti“

  I ) Výsledkem je *plán předávání* (release plan)

## *ad 3) Konstrukce*

- každá iterace je miniprojektem (analýza, návrh, kódování, testování, integrace), výsledkem ukázka zákazníkovi a systémový test
- redukce rizik prioritním řešením rizikových UC
- při iteracích se přepisuje část kódu (zabránění zhoršování)

● využití UML při konstrukci:
UC → rozsah (konceptuální diagram tříd) → návrh - jak budou třídy spolupracovat (CRC štítky, diagramy interakce) → kódování (s návratem v případě potřeby) → zdokumentování vytvořeného software (UML diagramy – diagram seskupení, rozmístění, diagram tříd specifikačního pohledu pro každé seskupení (nemusí být detailní), diagramy interakce, případně stavové diagramy, pro složité algoritmy popis strukturovaným jazykem, případně diagram aktivit + detailní dokumentace generovaná z kódu (např. JavaDoc))

## *ad 4) Přechod*

- postupné předávání zákazníkovi (např. mezi beta-verzí a konečnou verzí)
- některé činnosti nedělat brzy (optimalizace)
- neměl by probíhat vývoj s cílem přidat funkčnost (jen malé úpravy, resp. nezbytná funkčnost), jen pro opravu chyb