KIV/ASWI 2007/2008

Metody získávání a zachycení požadavků

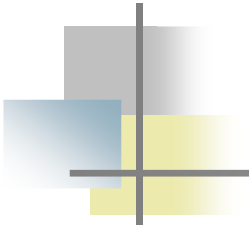
Postupy a UML modely

- Zachycení požadavků
- Model užití
- Popis problémové oblasti

Objektová analýza a návrh

- Sběr požadavků, analýza: *co* se chce
  - objektový model reálného/projektovaného světa
- Návrh: *jak* to realizovat
  - třídy a mechanismy pro efektivní realizaci

- Objektové metody
  - přirozený model, dobré zvládání složitosti
  - stabilita vzhledem k požadavkům
  - vnitřní kvalita, flexibilita

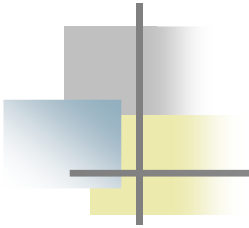


Typy požadavků

- Business reqts
  - » Vize a rozsah projektu
- User reqts, Business rules, Extra-func
  - » Use-cases
- System reqts, Functional reqts, Constraints
  - » Software Requirements Spec

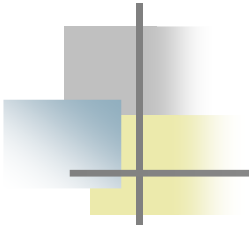
Postup práce s požadavky

- Reqts development
  - Elicit
  - Analyze, Negotiate
  - Document
  - Review
    - » Baselined requirements
- Reqts management
  - Change management



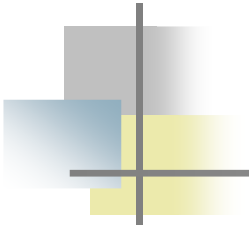
Lidé v analýze

- Zákazník
  - externí, interní
  - doménový expert
- Zainteresovaný hráč (stakeholder)
  - ředitel / investor / standardizační orgán / daňový poplatník
  - vliv na úspěch projektu
- Analytik



Úloha a vlastnosti analytika

- Hlavní úloha
  - zprostředkovatel mezi zákazníkem a programátory
- Dovednosti a vlastnosti
  - komunikační schopnosti, naslouchání a pozorování
  - vedení schůzek, organizační
  - schopnost abstrakce, nadhled, tvořivost
  - detailní znalost problémové oblasti
  - psaný projev, modelování



Obsah a cíl zachycení požadavků

- Jak popsat zadání tak, aby se z toho dalo vycházet pro implementaci, resp. jak umět číst takový popis
- Srozumitelnost pro zákazníka/analytiky, jednoznačnost a struktura pro návrháře/programátory/testery

Požadavky na software (software requirements)

- Požadavek = schopnost nebo vlastnost, kterou má software mít, aby jej uživatel mohl používat k vyřešení problému nebo dosažení cíle, který vedl k zadání, nebo aby splnil podmínky stanovené smlouvou, standardem, nebo jinou specifikací.
  - požadavkem není to, co uživatel nepotřebuje
  - požadavky jsou omezeny vnějšími podmínkami

- Typy požadavků
  - funkce (*functional reqts*)
  - vlastnosti (*non-functional reqts* ; mimofunkční pož.)

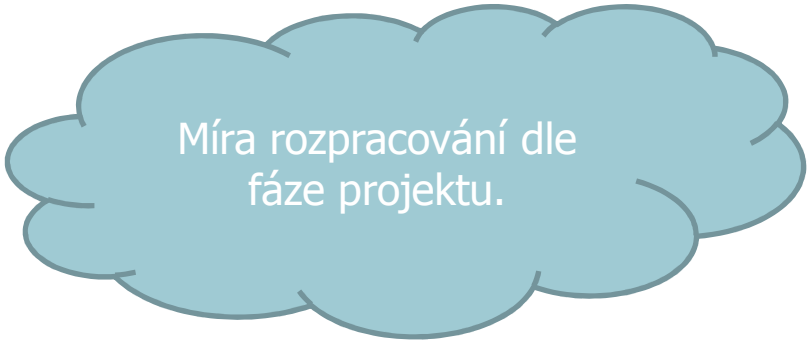
Techniky sběru požadavků

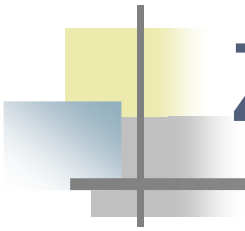
- Analýzy trhu, marketingové průzkumy

- Pozorování, práce s uživateli
- Rozhovory s uživateli
- Dotazníky
- Studium dokumentace, stávajících systémů
- Studium hlášení problémů (support, bug tracking)
- Předvedení prototypu

Formy zápisu požadavků

- Granularita, “dress code”
- Textový popis
  - shopping list, karty
  - strukturovaný text
  - IEEE normy
- Grafické notace
  - případy užití
- Implementace
  - prototyp
  - uživatelská příručka





Základní obrysy požadavků

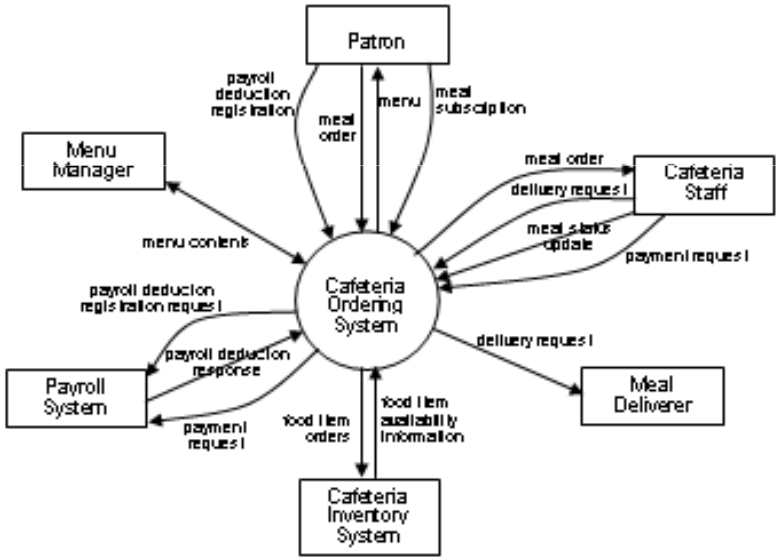
Stručný popis problému

- Základní, stručný popis účelu
  - vyjádřit cíl projektu
  - zabránit divergování během vývoje, nárůstu požadavků
- 25 slov / jeden odstavec
  - k čemu má systém sloužit
  - jaké informace bude udržovat
  - kdo jej bude používat
  - co přinese, čemu pomůže

Alternativa: RUP
- problém …
- postihuje … [koho]
- což vede k … [důsledky]
- řešení bude … [cílový stav]

Kontextový model

- Zobrazení vztahů systému s okolím
  - systém jako černá skříňka
  - aktéři, stakeholders
  - ostatní systémy



- Rozsah systému (kontrola *feature creep*)
- Rozhraní na okolí (HCI, API, data)

Vize produktu a rozsah projektu

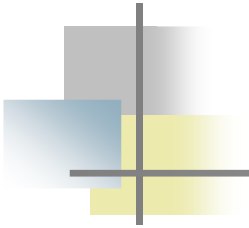
- Analýza: *co* se chce
- Vize: *proč* se to chce
  - pohled všech zainteresovaných účastníků
  - „problem behind the problem"
  - zdůvodnění výhodnosti projektu

- Přehled stakeholders a uživatelů
- Popis problému, obchodní příležitosti
- Kdo jsou zájemci o systém, potenciální konkurence
- Přehled očekávaných schopností a funkcí produktu
- Omezení, standardy, závislosti vztahující se k projektu

samostudium: šablona v RUP, Wiegers

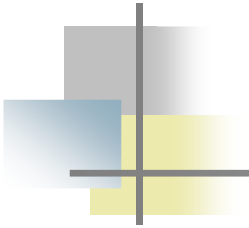
4(+1) kategorie požadavků

- Klasifikace funkčnosti do čtyř kategorií
  - jaké *informace* systém obsahuje, udržuje
  - jaké *funkce* poskytuje uživatelům
  - jaké *analýzy* dat provádí
  - jaké jsou *interakce* s jinými systémy
- Pro každý druh příslušné vlastnosti
  - stačí jednoduché seznamy
- To +1 je … *not this time* (až příště)
  - mimo rozsah zadání
  - doplňková funkčnost



Glosář

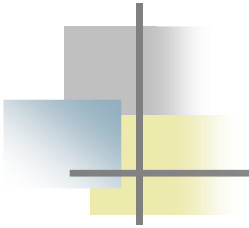
- Seznam důležitých pojmů
  - klíčové
  - nejasné
  - sporné
- Stručný, všemi odsouhlasený popis = společný slovník, prevence nedorozumění



Vlastnosti systému

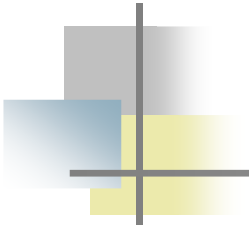
Je třeba vědět nejen *co*, ale také *jak dobře*.

- Co to je: mimo-funkční požadavky
  - doby odezvy, objemy, spolehlivost, …
  - nutný doplněk požadavků na funkce
  - dopad na architekturu, implementaci (někdy významný)



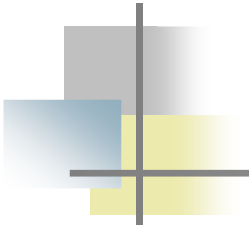
FURPS+

- Model třídění vlastností
  - Functionality, Usability, Reliability, Performance, Supportability + constraints
  - původně Hewlett-Packard
- Omezující podmínky
  - normy, zákony
  - obchodní pravidla
  - implementační omezení (technologie, rozhraní)
  - fyzické charakteristiky



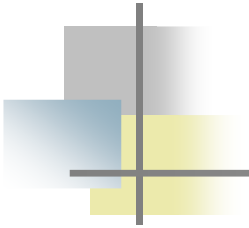
Reprezentace vlastností

- Účel: možnost ověřit splnění v implementaci
- Měřitelný způsob (numericky)
  - obvyklá hodnota, povolené odchylky / četnost / přírůstky
  - zvážení realizovatelosti funkčních požadavků
  - vhodná reprezentace pro neměřitelné
- Popis vlastností
  - u případů užití
  - u tříd problémové oblasti
  - vázané na celý systém



CRUD(L) matice

- Cíl: vědět kdo/co manipuluje s jakými údaji
- Create-Read-Update-Delete(-List)
- Úroveň detailu
  - analytická – uživatel, případ užití × informace
  - návrhová – třída, funkce, proces × tabulka

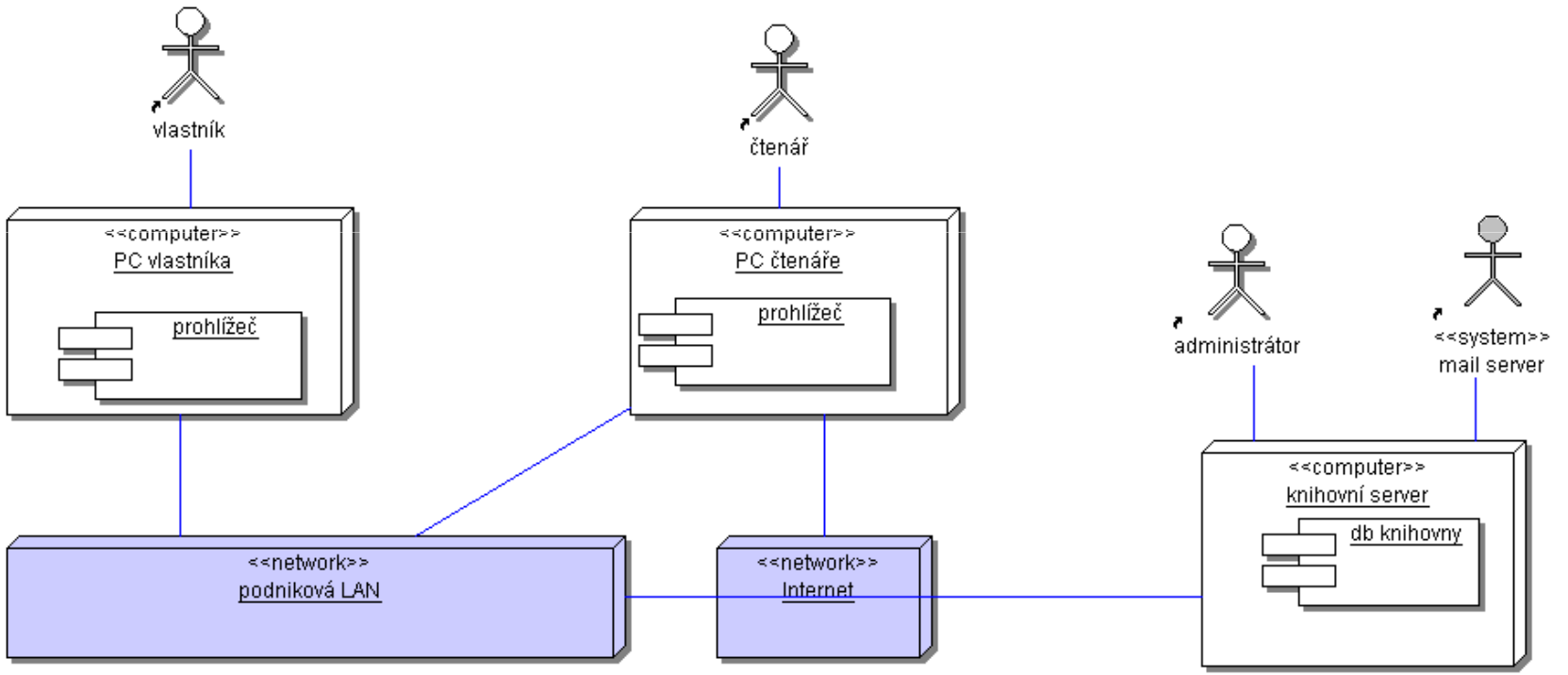


Fyzický rozsah systému

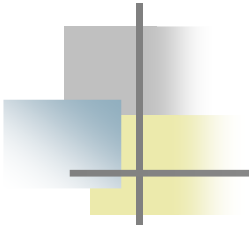
- Vztahy produktu k prostředí
  - porozumění run-time a fyzickému prostředí
  - odhad nákladů
- UML: model nasazení

- Alternativy
  - „zelená louka“ – součást návrhu architektury (později)

UML: diagram nasazení



Požadavky s UML: Model užití

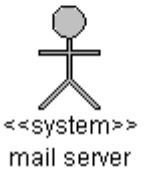


Přehled

- Popis požadavků na (vnější) funkčnost systému
  - Jací uživatelé k systému přistupují?
  - Co pro ně software dělá?
  - Jak systém zpracovává požadavky?
  - Kde je hranice systému (co je předmětem řešení)?
- Model = aktéři + případy užití
  - kontext, primární funkčnosti
  - další iterace, podružná funkčnost

Aktér

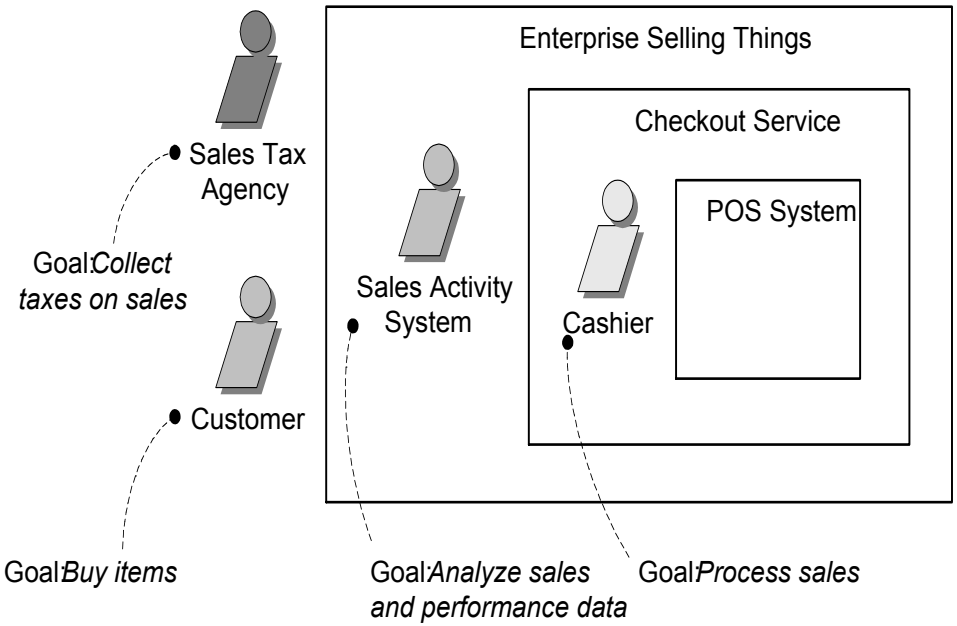
- Co to je: uživatel nebo jiný systém, který analyzovaný systém používá
  - typový uživatel, nachází se vně systému
  - spouští případy použití
  - primární / sekundární aktéři
- Popis aktéra
  - název: role, ne jména
  - jak a k čemu používá systém
  - seznam cílů
- Vazby mezi aktéry
  - generalizace – hierarchie rolí
    - » ne přístupová oprávnění



Jak najít aktéry

- Poznají se tak, že
  - odrážejí způsoby používání aplikace
  - jsou podstatní pro určení hranice systému

- Pozor
  - primární × sekundární
  - vytřídit *uvnitř systému* a *not this time*
  - některé budou nalezeny až později



Případy užití

Co to je případ užití (PU):

sekvence akcí, které systém provádí
v důsledku nějakého vnějšího podnětu
a které vedou k výsledkům
viditelným pro některého jeho uživatele.

- též "prototypová úloha" či „elementární business proces"
- poprvé Jacobson 1992 (OOSE)

Vlastnosti případů užití

- Obvykle jeden (primární) aktér
- Kompletní, uživatelsky užitečná funkce
  - často složené z více "transakcí"
  - PU je třída $\Rightarrow$ instance případu užití
- Realizace
  - pro aktéra nezajímavá (akce provádí systém)
  - vyjádření v UML (analýza/návrh): třídy, spolupráce, sekvence, stavy

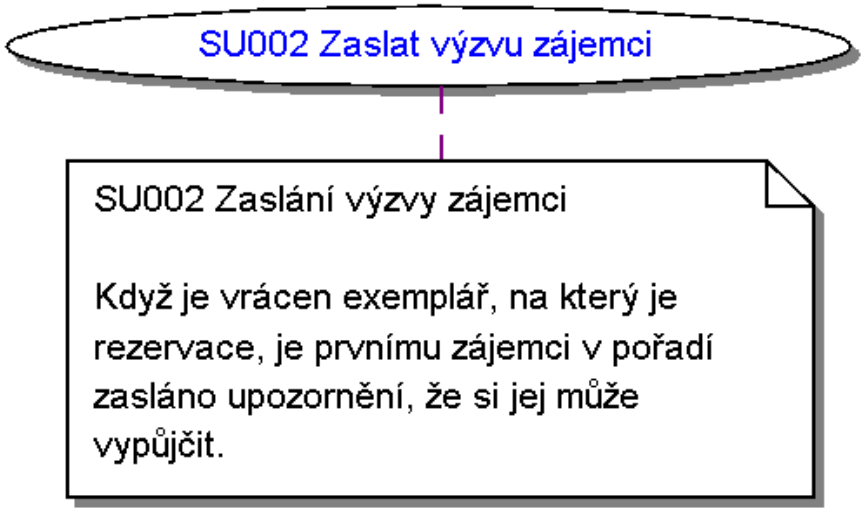
Jak najít případy užití

- Hledáme dialogy aktér–systém
- Začít od aktérů
  - popis problému z pohledu 1 aktéra
  - seznam cílů/potřeb aktéra => první diagram PU
  - "Jaké jsou hlavní akce, které provádí?"
  - "Jak vkládá / získává informace?"
  - "Potřebuje vědět o stavu systému?"
- Výsledek
  - seznam případů užití (názvy)

Základní popis případu užití

… ve fázi shromažďování požadavků:
základní popis dané funkce aplikace

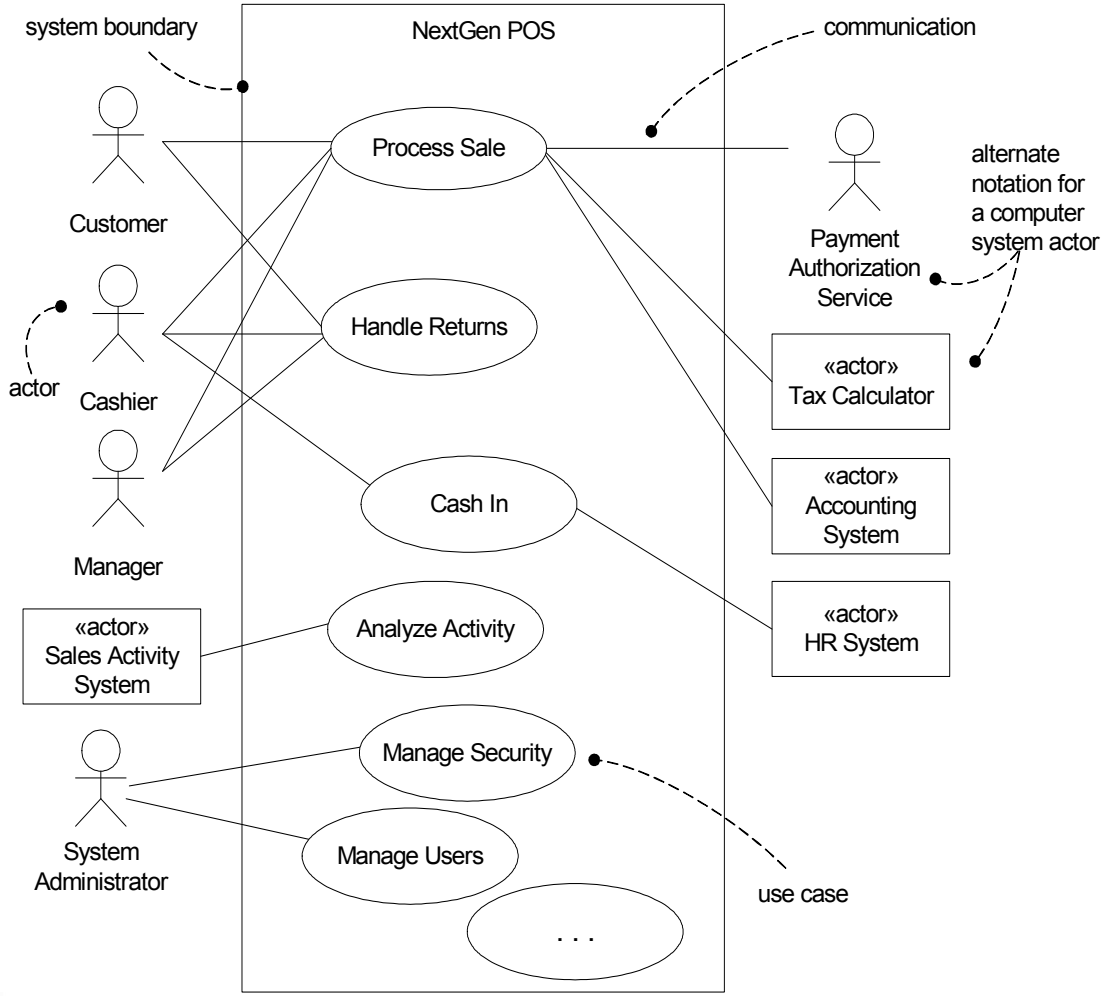
- Název
- Stručný popis účelu
- Základní kroky postupu
- Odkazy na zdrojové informace



Texty v případech užití

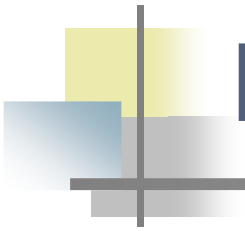
- Mluvit jazykem uživatele
  - konzultace s reálnými reprezentanty uživatelů
  - používat jeho termíny (glosář)
  - abstrahovat od možné implementace
    - » pozor na výrazy typu *okno*, *formulář*, *dialog*
- Průběh akcí psát hutně, srozumitelně
  - definovat spouštěcí událost
  - odrážky/číselné seznamy optimální
  - diagramy až v druhém kroku

UML – diagram případů užití

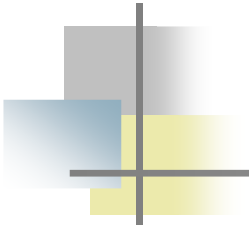


Granularita modelu

- Faktory: velikost systému, potřeby programátorů
- Subsystémy → typy funkčnosti, balíky PU, proces
  - » nikoli include/extend!
- Funkčnost → diagram, jednotlivé PU
- Pracovní úkoly → části popisu PU

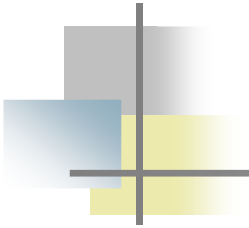


Detaily požadavků



Detailní analýza požadavků

- Fáze rozpracování
- Známe
  - zadání a cíl projektu
  - dobrá představa o klíčových funkcích, přehled o celku
  - možné zdroje problémů
- Co dál
  - doplnit detaily požadavků, kde je třeba
  - najít business mechanismy
  - vymyslet třídy pro jejich realizaci



Určení detailů PU

- Najít všechny (důležité) scénáře
  - vyjasnění nejednoznačností v zadání
  - detaily normálního průběhu
  - všechny alternativy
- Určit výsledný stav systému
- Očekávat změny
  - primární, sekundární PU
  - společná funkčnost $\rightarrow$ vkládané, zobecněné PU
  - změny v aktérech, sekundární aktéři

Podrobný popis případu užití

Detailní rozbor komunikace aktér-systém

① Standardní průběh
- nejčastější sled akcí
- bez chyb a různých možností

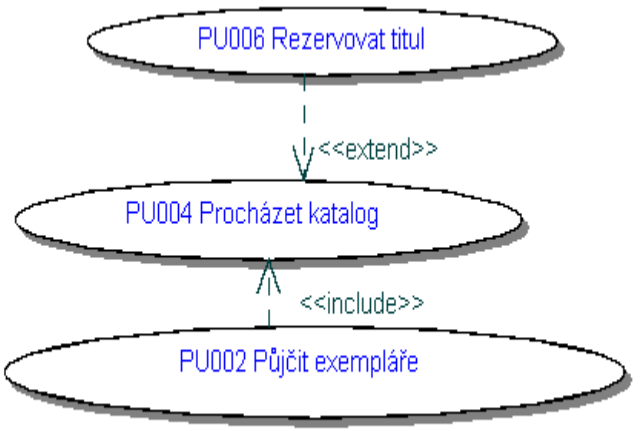
② Vstupní a výstupní podmínky
- co potřebujeme pro standardní průběh

③ Chybové stavy a alternativy
- zjištění, určení míst výskytu, příčin, následků
- popis alternativních a chybových akcí

```
Název a popis:

 PU002 Půjčit exempláře

 Umožňuje vlastníkovi zaevidovat vypůjčení exemplářů


Standardní průběh:

 # vlastník zvolí volbu "výpůjčka" v nabídce
 # čtenář oznámí vlastníkovi svoji identifikaci (jmé
 # vlastník zadá nebo vyhledá čtenáře v seznamu zamě
 <alt: čtenář nenalezen v evidenci>
 # systém zobrazí všechny volné exempláře vlastníka
 # pro všechny půjčované exempláře
 ## vlastník vyhledá vypůjčovaný exemplář ve svém fo
    podle PU004 Procházet katalog -- omezeno na fond
 ## systém ověří, že vybraný exemplář je k dispozici
    rezervovaný)
 <alt: na exemplář je rezervace>
 ## systém zobrazí návrh výpůjčky s datumem vrácení
 ## vlastník může data návrhu opravit, poté návrh od
 ## systém vytvoří záznam o výpůjčce exempláře čtená
    jeho data podle hodnot upravených vlastníkem
 ## systém informuje vlasntíka o vytvoření výpůjčky
 ## vlastník předá exemplář čtenáři
 # tento PU končí volbou "ukončit půjčování" zvoleno


Alternativní průběhy:

 čtenář nenalezen v evidenci (krok 3)
 - systém upozorní vhodným hlášením, tento PU končí

 na exemplář je rezervace (krok 5)
 - systém to oznámí vhodným hlášením
 - tento PU pokračuje krokem 4 - další exemplář k pů


Vstupní podmínky:

 (žádné)

Výstupní podmínky:

 - exemplář je zapůjčen čtenáři
 - je zaevidována výpůjčka
 - pro exemplář je nastaven příznak "vypůjčen"
 - systém je připraven pro libovolnou další operaci
```

Vztahy mezi případy užití

- Zdroje vztahů
  - z povahy věci + snaha o zvýšení obecnosti
- *Zahrnutí* jiného případu použití
  - rozšiřuje funkčnost, reuse vkládaného
  - kam vložit: uvést u vkládajícího
- *Rozšíření* případu použití o jiný
  - původní průběh nedotčen
  - zpracování nestandardních situací apod.
  - kam vložit: uvést u vkládaného
- *Generalizace* základního průběhu
  - specializované PU doplňují detaily



Výsledný model užití

- Úplnost
  - všichni aktéři
  - popis: 20% případů na 100% … 20% na 0%
  - pozor na „samozřejmosti" a skryté požadavky
  - včetně technických PU a re-use scénářů

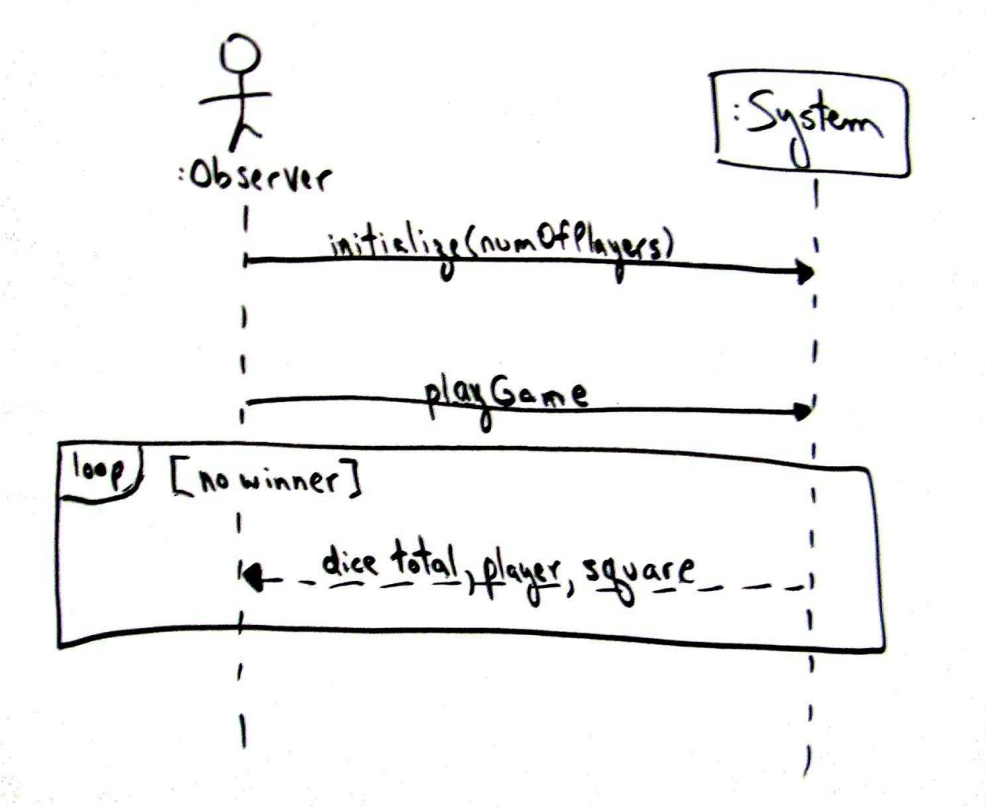
- Přehlednost a srozumitelnost
  - rozložení diagramu
  - hodně PU $\Rightarrow$ více diagramů
    - klíčová a doplňková funkčnost
  - texty srozumitelné pro zákazníka
    - schvalování, kontrakt

Další modely požadavků

- Obecný cíl = zachycení „jak funguje doména“
  - » stejné pro případy užití
- Často používané modely
  - aktivit – business procesy
  - stavový – uživatelské rozhraní, entity
- Další podpůrné nástroje
  - prototyp
  - formální specifikace

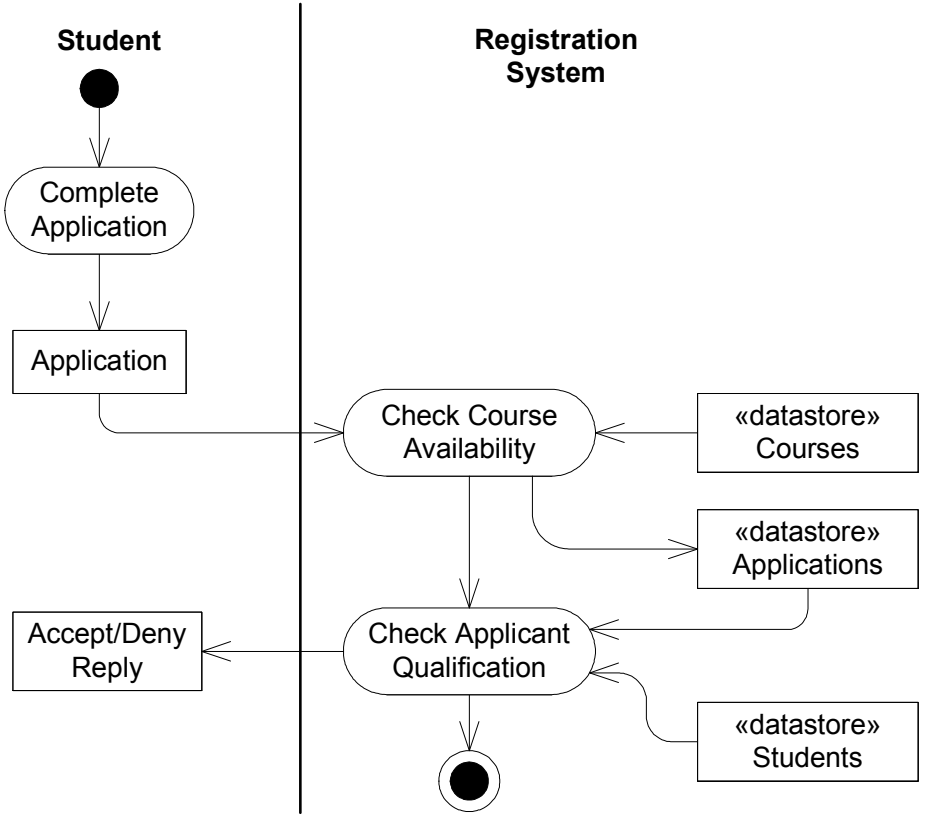
Systémové sekvenční diagramy

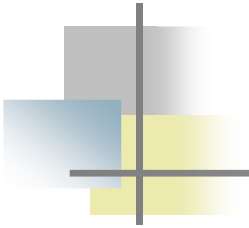
- shrnutí komunikace aktér-systém



Procesní modely

- Popis scénáře PU při složitém rozhodování
  - » když text nepřehledný
- Nadřazený proces dělený do PU
  - » business process model
- Notace
  - UML diagram aktivit
  - DFD



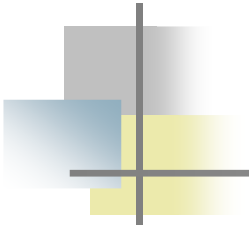


Chování aplikace

- Typicky chování UI
  - správa dat, průvodci, přihlašování, ...
  - náhrada modelu uživatelského rozhraní
- Prototyp uživatelského rozhraní
  - papír → PPT → demo
- Stavový model
  - stav=obrazovka / krok průvodce / ...

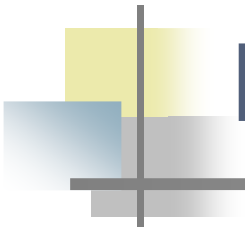
Prototyp uživatelského rozhraní

- **Prototyp** = reprezentace vnějšího rozhraní produktu nebo jeho části, v abstraktní (např. papírové) či konkrétní (spustitelná aplikace) podobě.
  - podklad pro určování funkcí a ovládání aplikace
  - diskuse nad prototypem $\Rightarrow$ korekce neshod v porozumění
- Diskuse k prototypování
  - jednoduchá testovací data
  - abstraktní podoba uživatelského rozhraní
  - řízení konečným automatem
- Kam s ním?
  - vyhodit $\times$ uchovat
    - hraniční třídy pro objektový návrh



Formální popisy

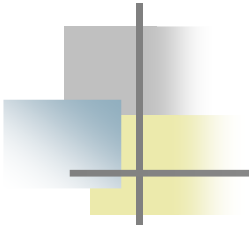
- Matematický model chování aplikace/systému
- Dokazatelná
  - správnost, bezrozpornost
  - úplnost
- Specializované notace
  - Z, VDM
  - B-method
- … a nástroje



Doménový a datový model

Doménový model

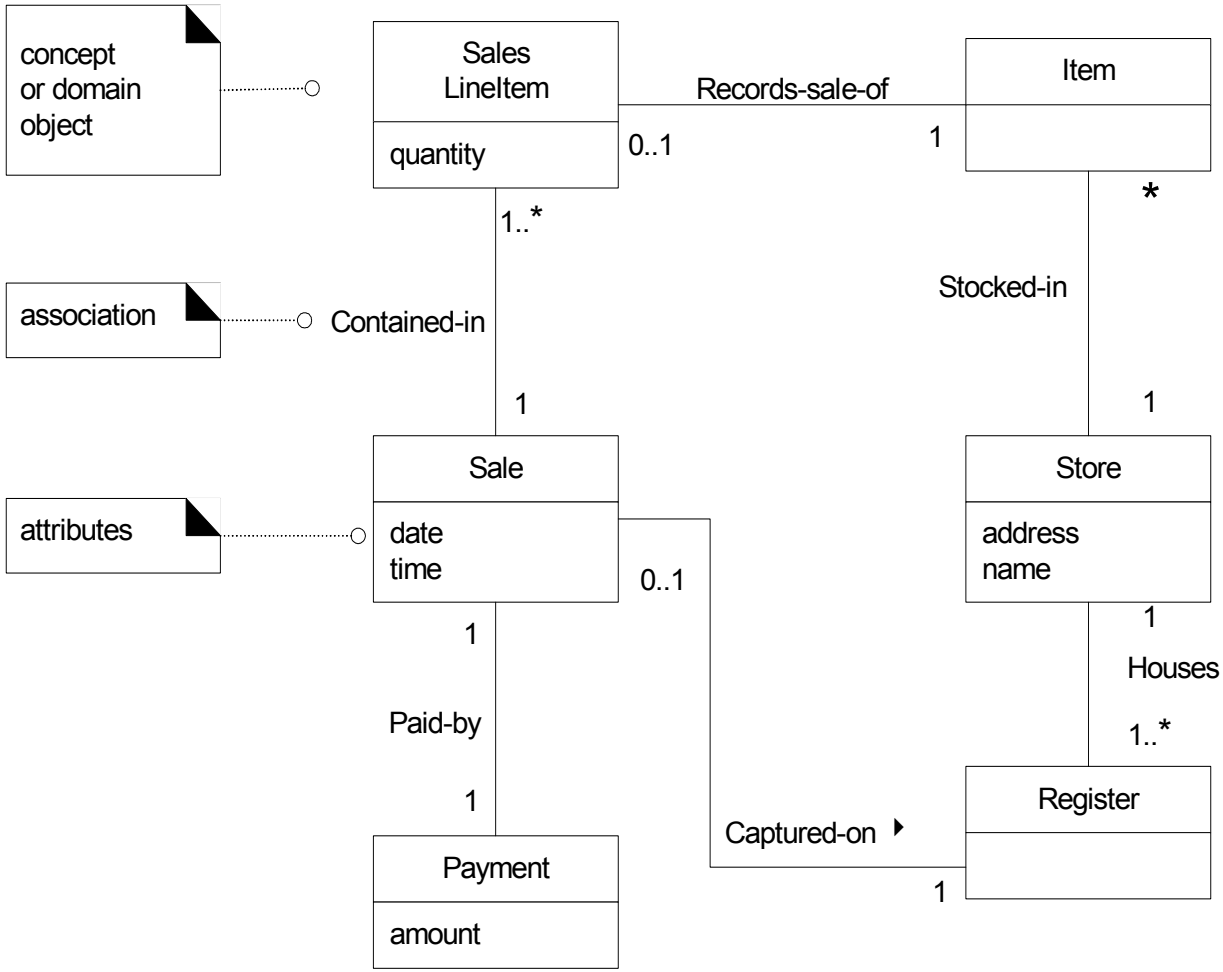
- Popis struktury problémové oblasti
  - Jaké jsou základní abstrakce používané v oblasti aplikace?
  - Jaké mají názvy, vzájemné vztahy a vlastnosti?
  - Jakým postupem je získáme?
  - Podle čeho si máme vymyslet [stabilní] třídy pro realizaci?
- Východisko = glosář
- Model = doménové objekty (diagram tříd)



Doménové objekty/třídy

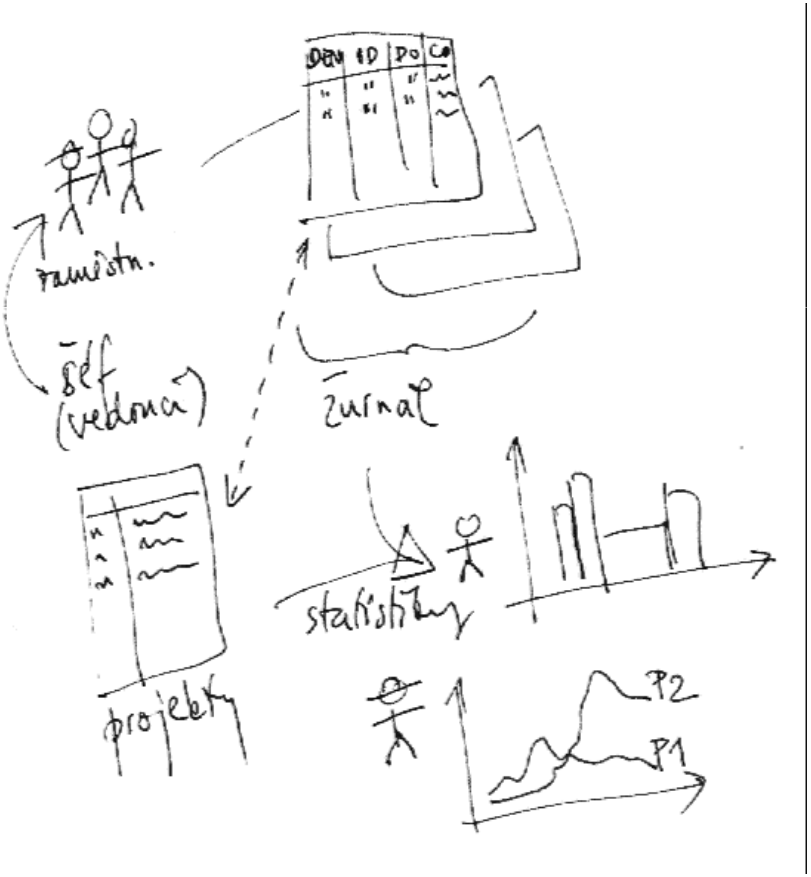
- Doménové objekty $\Rightarrow$ třídy
  - "věci" vyskytující se v problémové doméně
  - klíčové pro fungování systému
  - systém udržuje informace
- Podstatné aspekty
  - terminologie uživatele, pojmy $\rightarrow$ názvy tříd
  - jen základní obrysy
  - vztahy mezi třídami (asociace, kardinality)
  - nezávislost na implementaci

UML: doménový diagram tříd



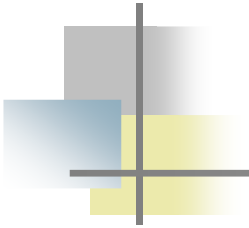
Jak najít doménové objekty

- Doménová analýza
  - konzultace
  - doménový expert
  - části systému podstatné z *jejich* pohledu
- Pomůže
  - obrázek
  - rozhovor s uživatelem
  - pozorování práce



„Naivní“ a heuristické metody

- Analýza slovního popisu systému
  - podstatná jména ⇒ objekty/třídy
  - slovesa ⇒ chování, metody
  - atributy ⇐ podst. jména, příd. jména, …
  - závislá na kvalitě a rozsahu textu
- Předměty a role v dané oblasti
  - hmotné předměty, zařízení
  - místa, prostory, soubory míst (byt – dům)
  - koncepty, zaznamenávané události
  - role lidí

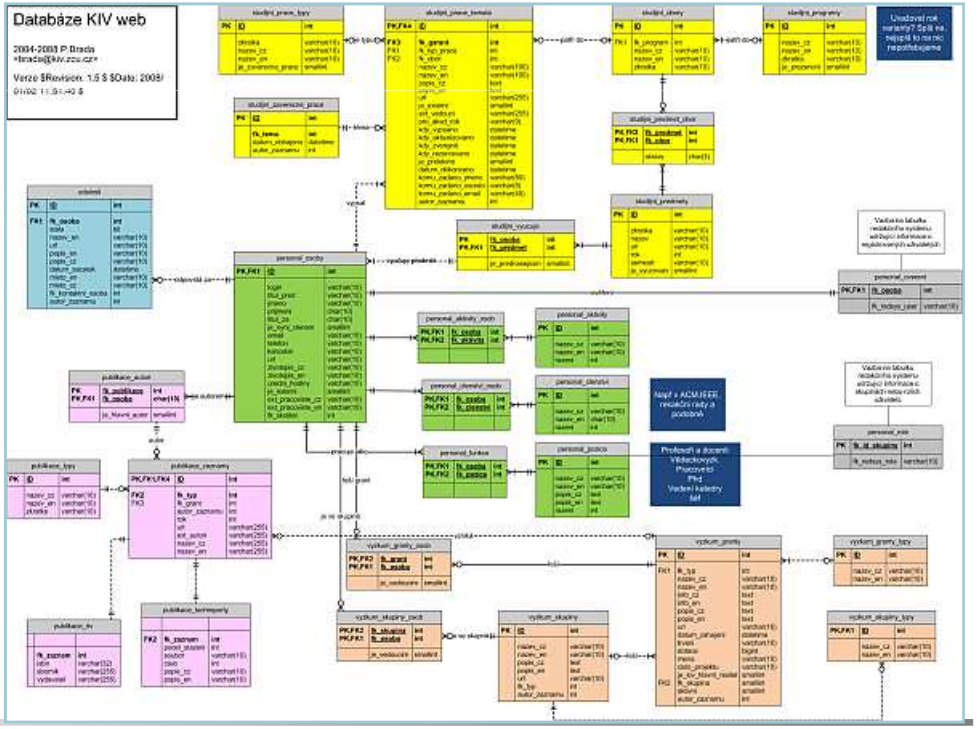


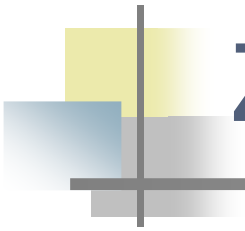
Použití doménového modelu

- Komunikace
  - dorozumění s klientem
- Objektová analýza a návrh
  - s těmito třídami můžeme najisto počítat
  - analýza přidá zodpovědnosti, detaily vlastností a chování
  - návrh je transformuje a přidá implementační třídy
- Datové modelování
  - doménový model → logický datový model

Datový model

- Logický (konceptuální), fyzický
  - » obvykle jeden z prvních nezávisle na OO technikách
- Viz DB1/DB2/PSDS





Závěrečné poznámky

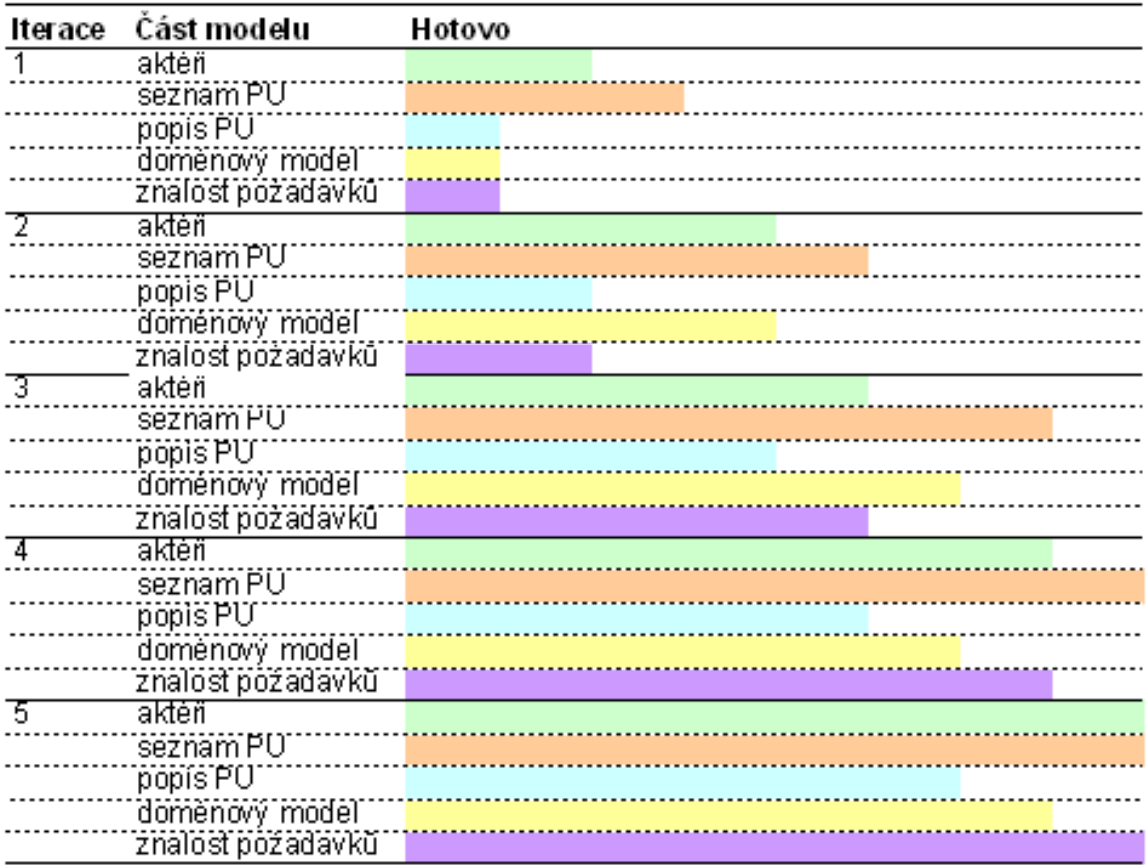
Specifikace požadavků a PU

- Model užití nemusí být (jedinou, úplnou) specifikací funkčnosti
  - funkcionality dle IEEE standardu DSP
  - story cards pro agilní metodiky
  - business procesy
  - systémové požadavky pro hw-sw systémy

Úplnost modelu požadavků

- Fáze zahájení projektu
  - přesně cíl/vize projektu
  - seznam klíčových aktérů, jejich cíle
  - seznam/diagram podstatných případů užití (dle cíle)
  - stručný popis klíčových PU, znalost klíčových vlastností
- Fáze rozpracování
  - kompletní seznam aktérů, popis důležitých
  - přesné diagramy užití, 100% PU
  - přesné popisy důležitých PU, stručné u všech
  - přesný popis vlastností

Vývoj znalosti požadavků



» V závislosti na iteraci, fázi, release

Shrnutí

- Prostředky UML a OOA pro zachycení požadavků
  - kontext – vize, aktéři
  - funkčnost – případy užití
  - struktury – doménový model
  - vlastnosti – u PU, doménových tříd, samostatně

- Poznávání požadavků je nekončící proces
  - zákazník a jeho představy se mění
  - poznání analytiků se zpřesňuje analýzou i implementací