Základní etapy vývoje

analýza, návrh, implementace, ...

© 2008 Radek Ošlejšek
FI MU Brno

oslejsek@fi.muni.cz
http://www.fi.muni.cz/~oslejsek/PA103

Základní etapy vývoje

- Většina metodik vývoje softwaru v sobě nějakým způsobem zahrnuje následující kroky:
  - Zachycení požadavků na systém
    - Externí modely (pohledy na systém)
  - Analýza
    - Základní modely, bez implementačních detailů.
    - Většinou platformě nezávislé
  - Návrh
    - Detailní modely
    - Zvolený cílový jazyk, technologie, ...
  - Implementace
  - Testování

Zachycení požadavků na systém

Zachycení požadavků

- Vstupem je často analýza obchodních procesů (*business processes*)
- Textová specifikace
- Analýza textů
- Digramy případů užití
  - diagram(y) + dokumentace

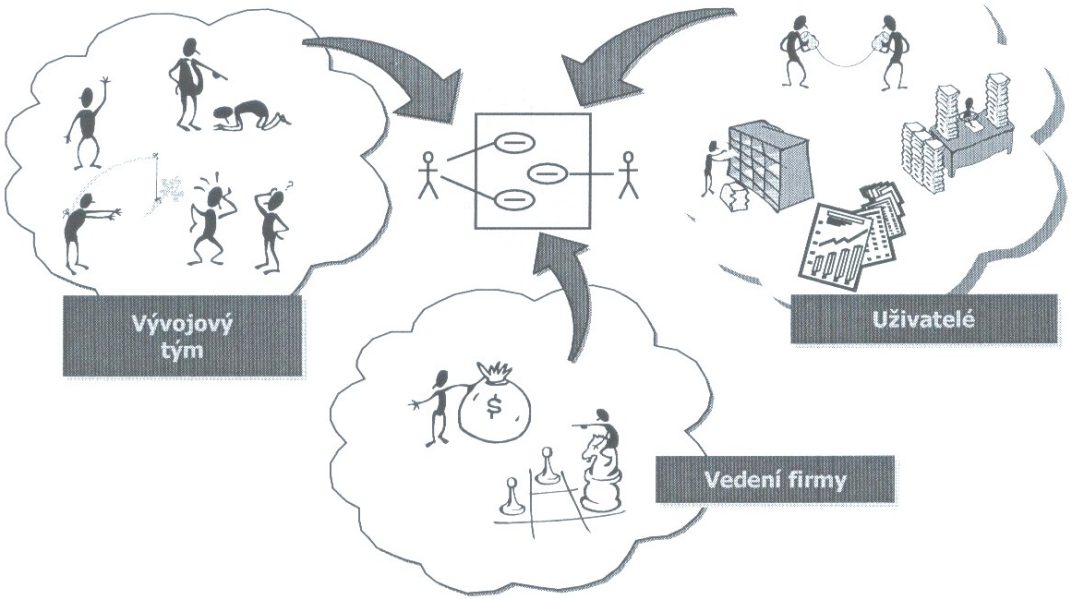
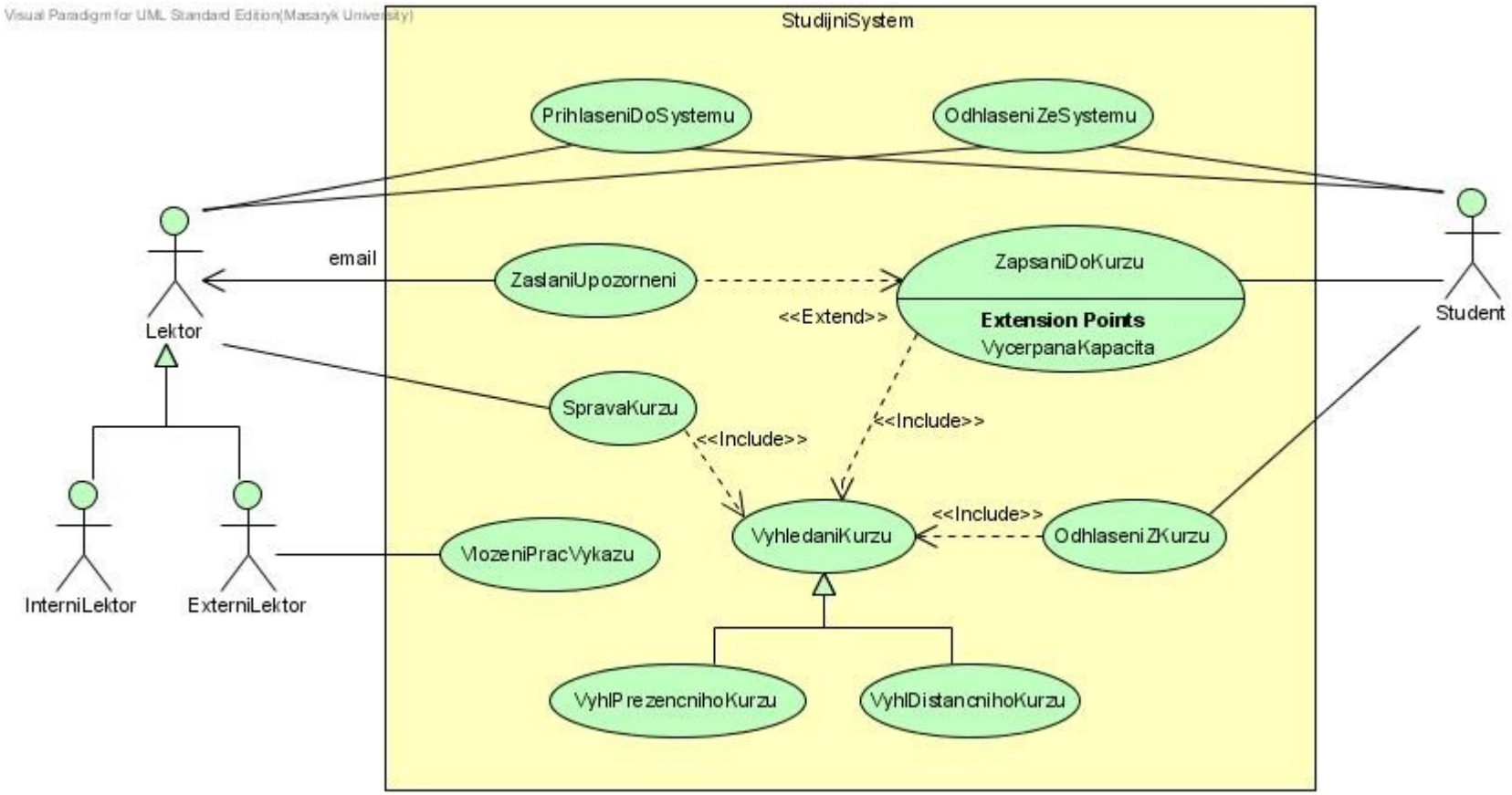


Diagram případů užití

Příklad



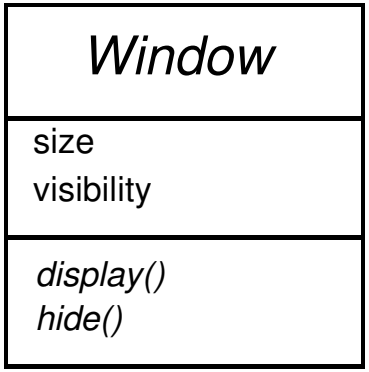
Analýza

Analýza – nejčastěji využívané diagramy

- Diagram tříd
  - Analytický model
    - Základní operace a atributy, dědičnost, asociace (většinou obousměrná)
    - Prvotní rozdělení zodpovědnosti
    - Asociační třídy, násobnost asociací M:N, asociace s kvalifikátorem,...
- Diagram balíků
  - Základní rozdělení do balíků
- Stavový diagram
  - Chování některých objektů/tříd
- Další diagramy dle potřeby

Analytický model tříd

- Zachycuje doménovou oblast, později jsou analytické třídy upřesněny/rozloženy na návrhové třídy
- Obsahují pouze klíčové atributy a operace nutné pro popis základní zodpovědnosti třídy
  - Jméno: je nezbytné
  - Atributy: pouze podmnožina budoucích atributů, typy se vynechávají
  - Operace: parametry a návratová hodnota jen pokud je to důležité
  - Viditelnost: nezobrazuje se



Vlastnosti dobré analytické třídy

- Pojmenování třídy odráží její význam
- Jasně definovaná množina zodpovědností
  - zodpovědnost = typ služby, kterou třída nabízí ostatním třídám
  - Př. zodpovědností pro analytickou třídu *NákupníKošík*:
    - OK: přidej položku do košíku, odeber položku z košíku, ukaž položky v košíku
    - také OK: spravuj položky v košíku
    - špatně: ověř kreditní kartu, akceptuj platbu, vytiskni účet
- Malý počet vazeb na ostatní třídy
  - pro řešení své zodpovědnosti vyžaduje třída spolupráci pouze malého počtu dalších tříd
  - rovnoměrné rozložení zodpovědností mezi třídy vede k malému počtu propojení

Nalezení analytických tříd

- Analýza podstatných jmen a sloves
  - analýza textů (specifikace systému, model požadavků, dokumentace případů užití, atd.)
  - podstatná jména v textu často vedou na třídy nebo atributy
  - slovesa a slovesné fráze často indikují zodpovědnosti nebo operace
  - pozor na synonyma a homonyma
- CRC analýza
  - class, responsibilities and collaborators analysis
  - “brainstorming” metoda s lepícími papírky
    - co papírek, to jedna třída
    - každý papírek obsahuje jméno, zodpovědnosti a spolupracující třídy
    - místo seznamu spolupracujících tříd se mohou nalepit na tabuli a propojit čárami
  - používá se spolu s analýzou podstatných jmen a sloves
- Další zdroje tříd
  - fyzické objekty (lidé, pracoviště,...), dokumenty (formuláře, objednávky,...), známá rozhraní (terminály, periferní zařízení), ...

Analýza podstatných jmen a sloves

Informační systém *Studium*

- Informační systém *Studium* bude sloužit pro podporu správy kurzů včetně elektronického přihlašování a odhlašování.
- Správa studijních kurzů umožní lektorům přidávání nových kurzů, mazání kurzů a úpravu stávajících parametrů, kterými jsou název, popis, prerekvizity, kapacita, nastavení rozvrhu a rozlišení, zda jde o prezenční nebo distanční kurz.
- Studenti dostanou možnost využít systém k elektronickému zápisu do kurzů a odhlašování z kurzů. Zápis do kurzu bude umožněn jen v případě, že dosud není naplněna kapacita kurzu. Pokud dojde k vyčerpání kapacity kurzu, systém pošle upozornění na email lektora zodpovědného za kurz.
- Jednou z doplňkových funkcí systému bude podpora pro vkládání pracovních výkazů o odpracovaných hodinách pro externí lektory, kteří na rozdíl od interních lektorů nepobírají měsíční mzdu a jsou placeni na základě odpracovaných hodin.

Legenda:

červeně - podstatné jméno kandidátem na třídu

zeleně - podstatné jméno, kandidátem na atribut

modře - sloveso kandidátem na operaci/metodu

CDC analýza

Visual Paradigm for UML Standard Edition(Masaryk University)

Kurz/PrezencniKurz/DistancniKurz	
Attributes:	
Name	Description
nazev	
prerekvizity	
popis	
kapacita	
rozvrh	
Responsibilities:	
Name	Collaborator
pridatKurz	zodpovedny Lektor
smazatKurz	zodpovedny Lektor
upravitUdaje	zodpovedny Lektor
prihlasitStudenta	Student
odhlasitStudenta	Student
zaslatUpozorneni	vyucujici Lektori

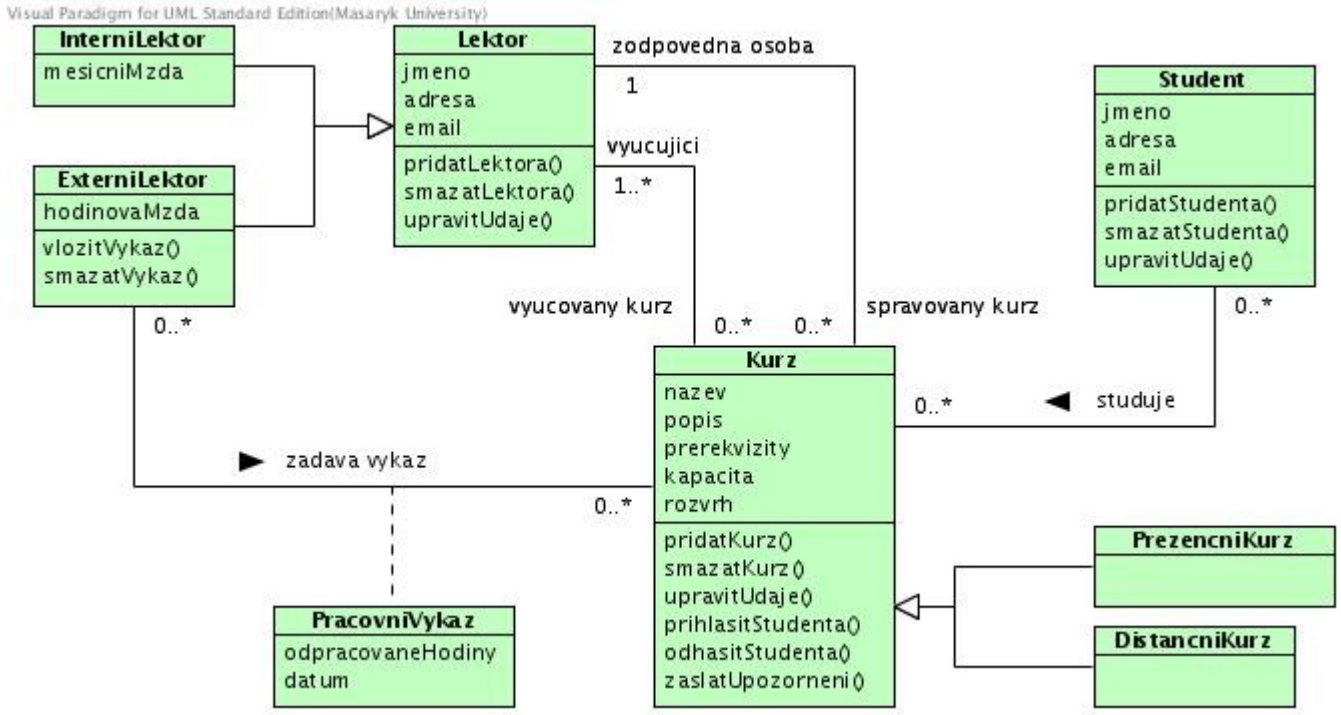
Student	
Attributes:	
Name	Description
jmeno	
adresa	
email	
Responsibilities:	
Name	Collaborator
pridatStudenta	Studijni
smazatStudenta	Studijni
upravitUdaje	Studijni

PracovniVykaz	
Attributes:	
Name	Description
odpracovaneHodiny	
datum	
Responsibilities:	
Name	Collaborator
vlozitVykaz	ExterniLektor, Kurz
smazatVykaz	ExterniLektor, Kurz

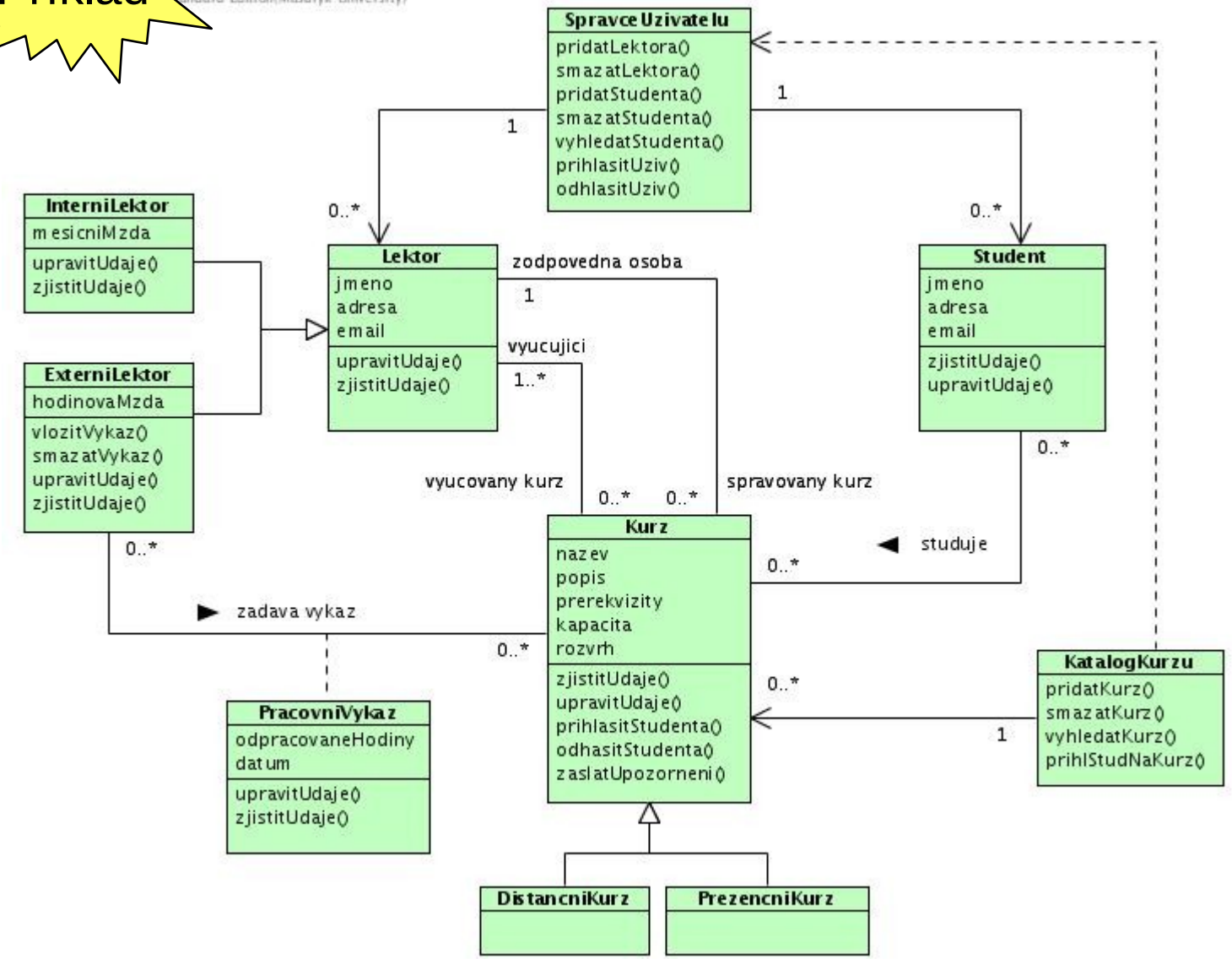
Lektor/InterniLektor/ExterniLektor	
Attributes:	
Name	Description
jmeno	
adresa	
email	
mesicniMzda/hodinovaMzda	
Responsibilities:	
Name	Collaborator
pridatLektora	Personalista
smazatLektora	Personalista
upravitUdaje	Personalista
vlozitPracVykaz	PracovniVykaz

Analytický model tříd

Příklad



Příklad

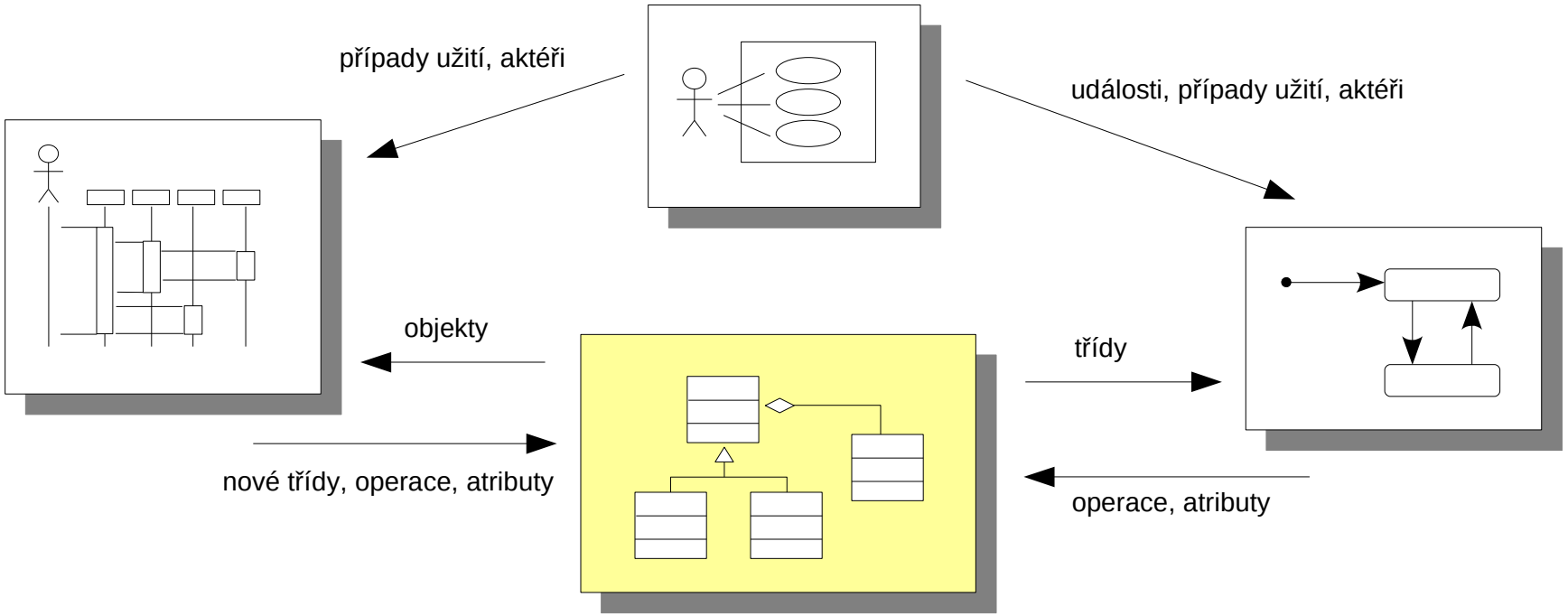

SpravceUzivatelu
pridatLektora()
smazatLektora()
pridatStudenta()
smazatStudenta()
vyhledatStudenta()
prihlasitUziv()
odhlasitUziv()
1
1
0..*
0..*
InterniLektor
mesicniMzda
upravitUdaje()
zjistitUdaje()
Lektor
jmeno
adresa
email
upravitUdaje()
zjistitUdaje()
zodpovedna osoba
1
Student
jmeno
adresa
email
zjistitUdaje()
upravitUdaje()
ExterniLektor
hodinovaMzda
vlozitVykaz()
smazatVykaz()
upravitUdaje()
zjistitUdaje()
vyucujici
1..*
0..*
vyucovany kurz
0..*
0..*
spravovany kurz
Kurz
nazev
popis
prerekvizity
kapacita
rozvrh
zjistitUdaje()
upravitUdaje()
prihlasitStudenta()
odhasitStudenta()
zaslatUpozorneni()
studuje
0..*
0..*
0..*
zadava vykaz
0..*
PracovniVykaz
odpracovaneHodiny
datum
upravitUdaje()
zjistitUdaje()
0..*
1
KatalogKurzu
pridatKurz()
smazatKurz()
vyhledatKurz()
prihlStudNaKurz()
DistancniKurz
PrezencniKurz

Návrh (design)

Návrh – nejčastěji využívané diagramy

- Diagram tříd
  - Návrhový model
    - přesně definované „malé“ implementovatelné třídy
    - modelování rozhraní
    - komplikovanější model
- Sekvenční a komunikační diagramy
  - Realizace (rozpracování) případů užití
  - Upřesnění tříd, upřesnění zodpovědností, nalezení nových tříd, operací, ...
- Diagramy balíků
  - Návrhové subsystémy
- Diagramy komponent
- Diagram rozmístění

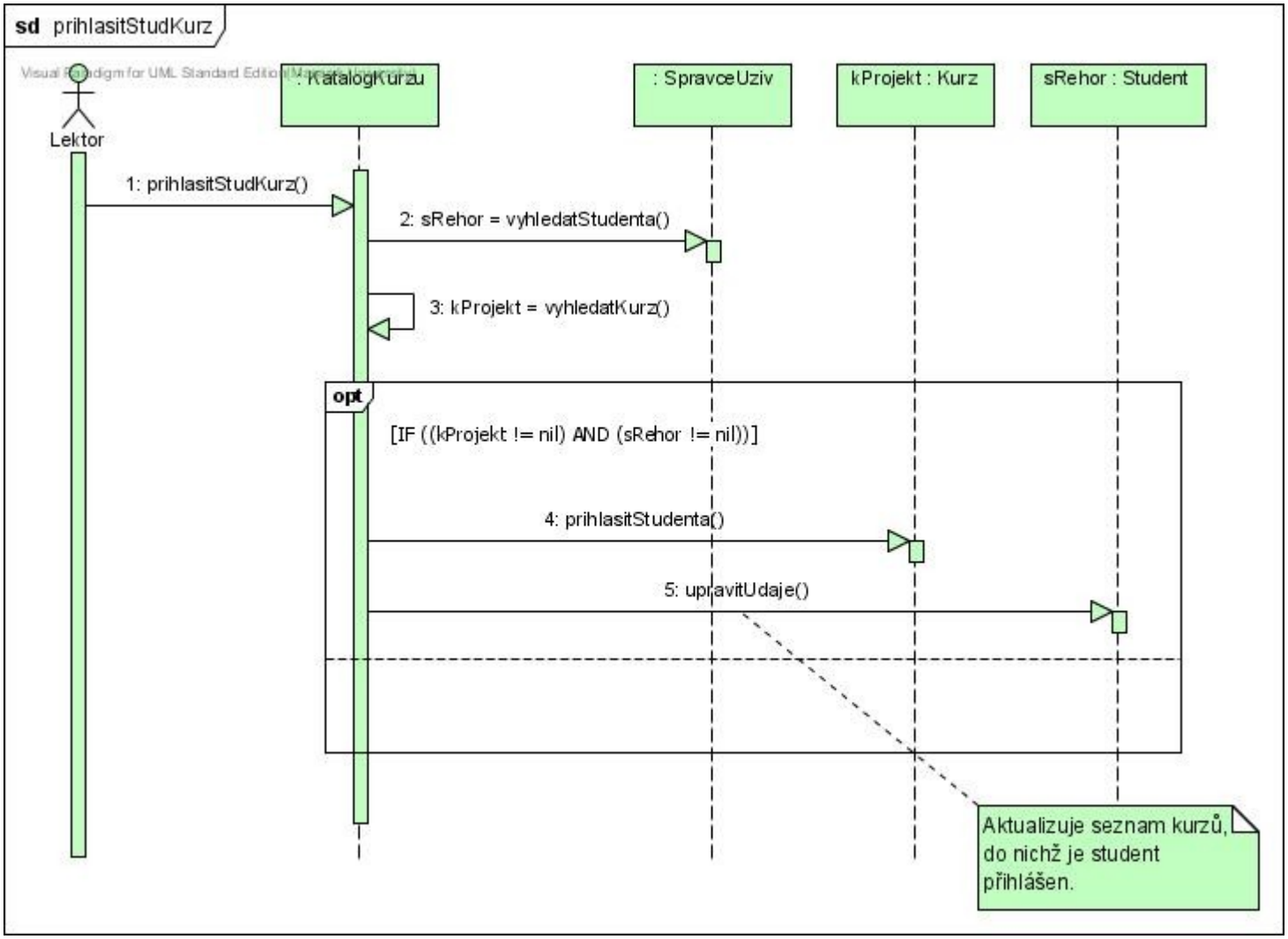
Integrace modelů



- Diagram tříd je základem
- Každý model přispívá něčím jiným
- Abstrakce a detail musí být konzistentní
- Integrace prostřednictvím *repository*
  - podpora CASE nástrojů pro udržení konzistence

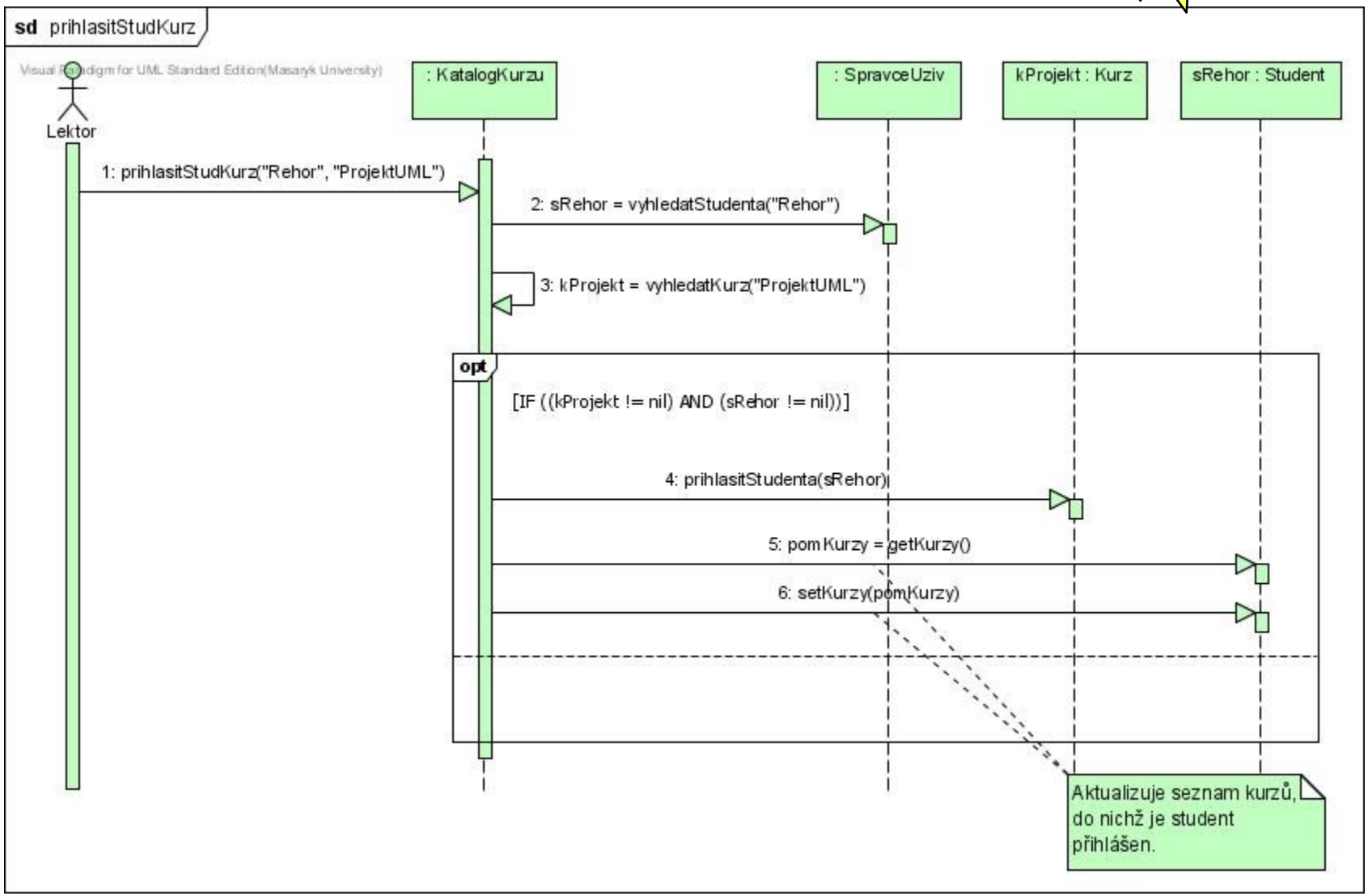
Sekvenční diagram (I)

Příklad



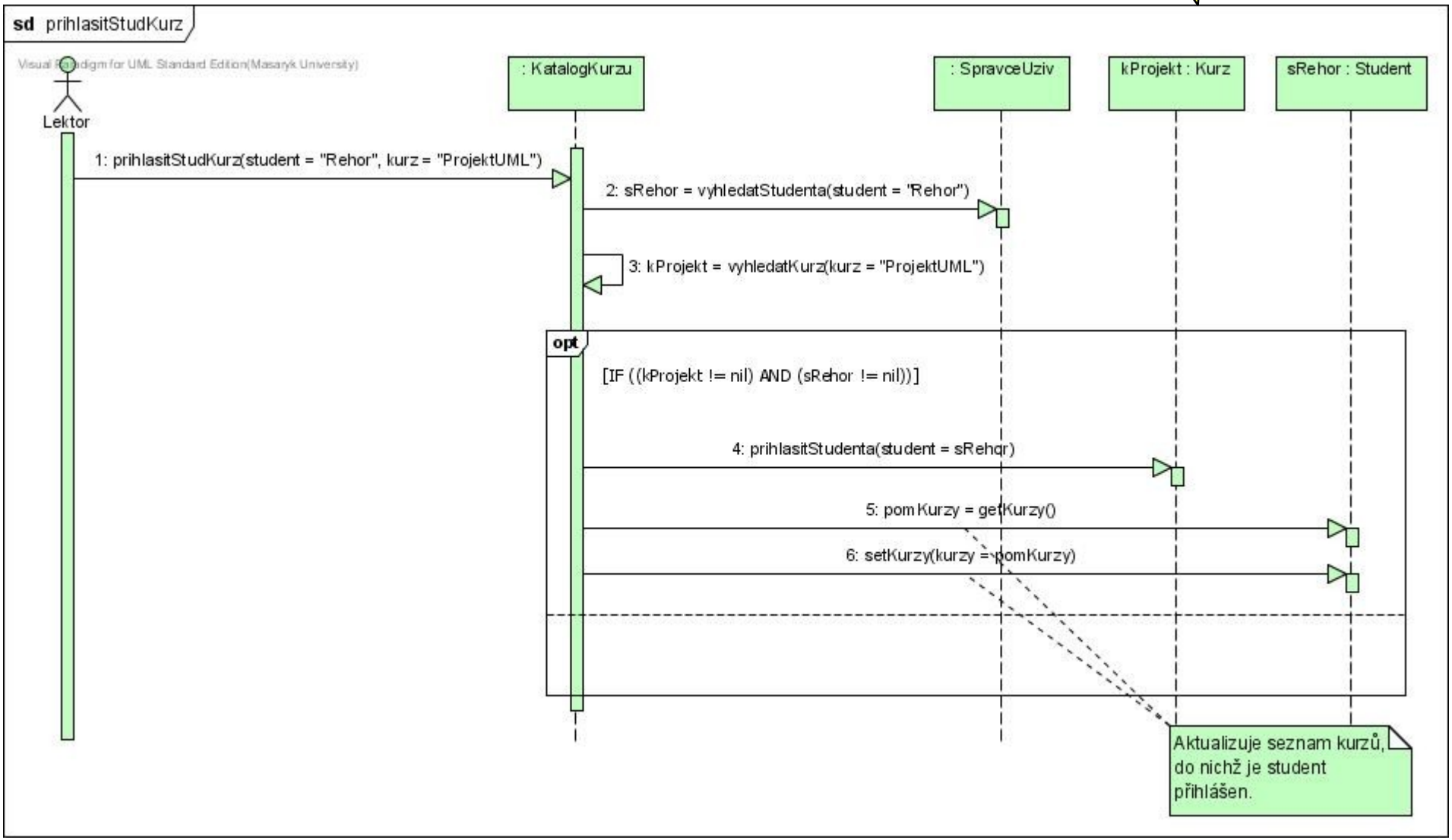
Sekvenční diagram (II)

Příklad



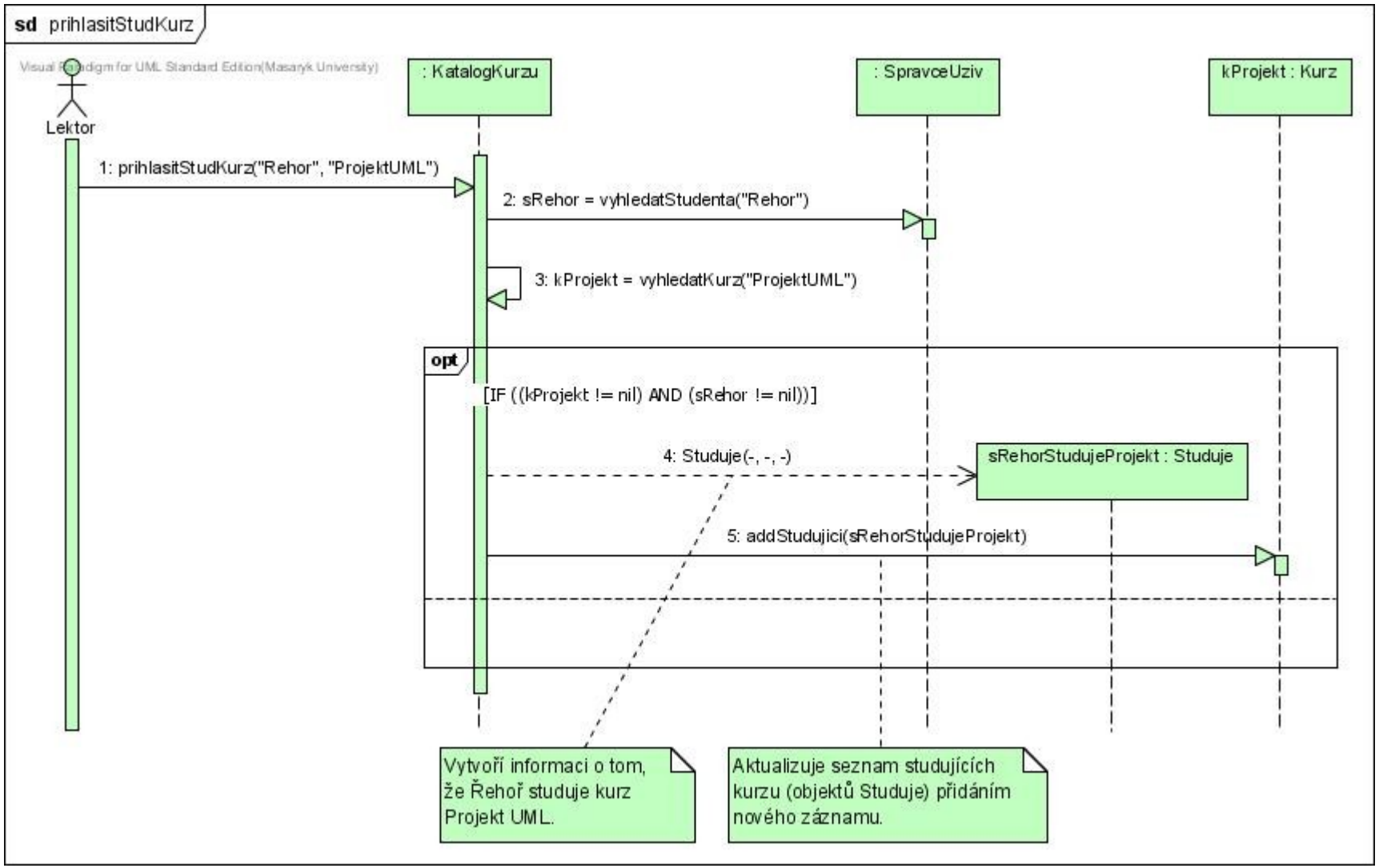
Sekvenční diagram (III)

Příklad



Sekv. diagram (IV) - alternativa

Příklad



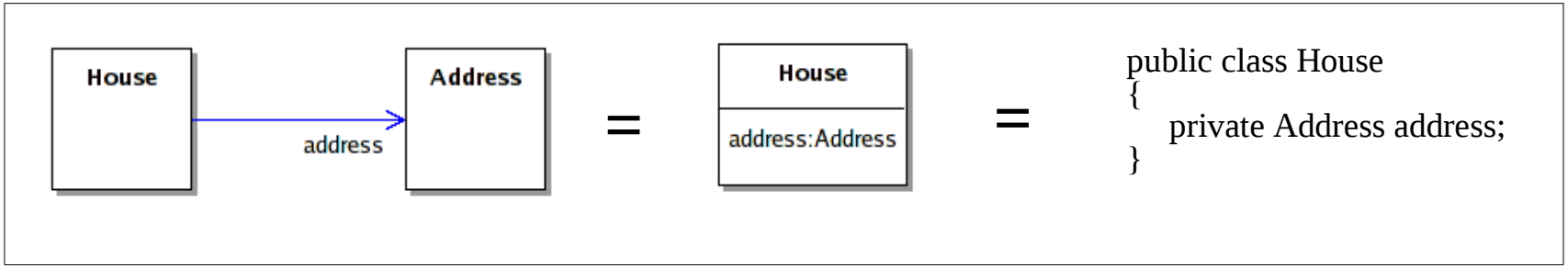
Návrhový model tříd

- Často pro konkrétní cílový jazyk a technologie
- Obsahuje všechny nezbytné detaily
- Úplnost
- Jednoduchost
- Vysoká soudržnost (koheze, zodpovědnost)
- Co nejmenší propojení s ostatními třídami
- Upřesněné asociace
  - Směr, agregace/kompozice, kardinalita, ...

- Dědičnost vs. asociace
- Násobná dědičnost
- Dědičnost vs. realizace rozhraní

Asociace a atributy

- Mezi asociacemi tříd a atributy tříd existuje velmi těsná vazba.
- Vazba 1:1 odpovídá odkazu atributem
- Vazba 1:N odpovídá odkazu polem atributů nebo jedné kolekci
- Vazba M:N se řeší v návrhu dekompozicí
- Směr asociace udává, která třída obsahuje atribut
- Obousměrné asociace se upřesňují při návrhu
- Role v asociaci se často stává základem pojmenování atributu



Směr řízení v asociaci

Je nutné se předem dohodnout na jedné z následujících interpretací:

1) Explicitní navigace (striktní interpretace podle UML2):
   - všechny šipky a křížky jsou ukázány, pokud chybí, je směr asociace (zatím) nedefinovaný
2) Potlačená navigace:
   - žádné šipky a křížky nejsou ukázány,
   - moc se nepoužívá – zakrývá příliš mnoho informace
3) Běžná praxe:
   - křížky se nepoužívají, obousměrné asociace nemají žádné šipky, jednosměrné asociace mají jednu šipku

Explicitní navigace

A komunikuje s B
komunikace B->A: není definováno

Běžná praxe

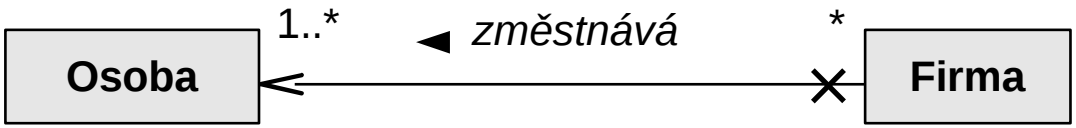
A komunikuje s B
B nekomunikuje s A

komunikace A->B: není definováno
komunikace B->A: není definováno

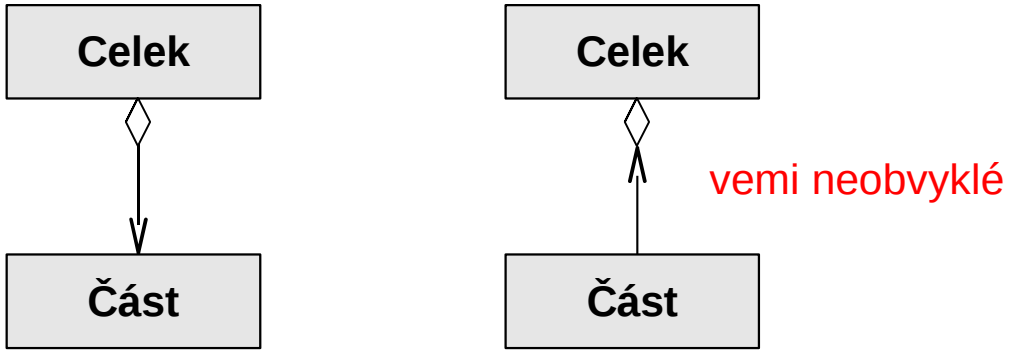
A komunikuje s B
B komunikuje s A

Směr řízení v asociaci – pokr.

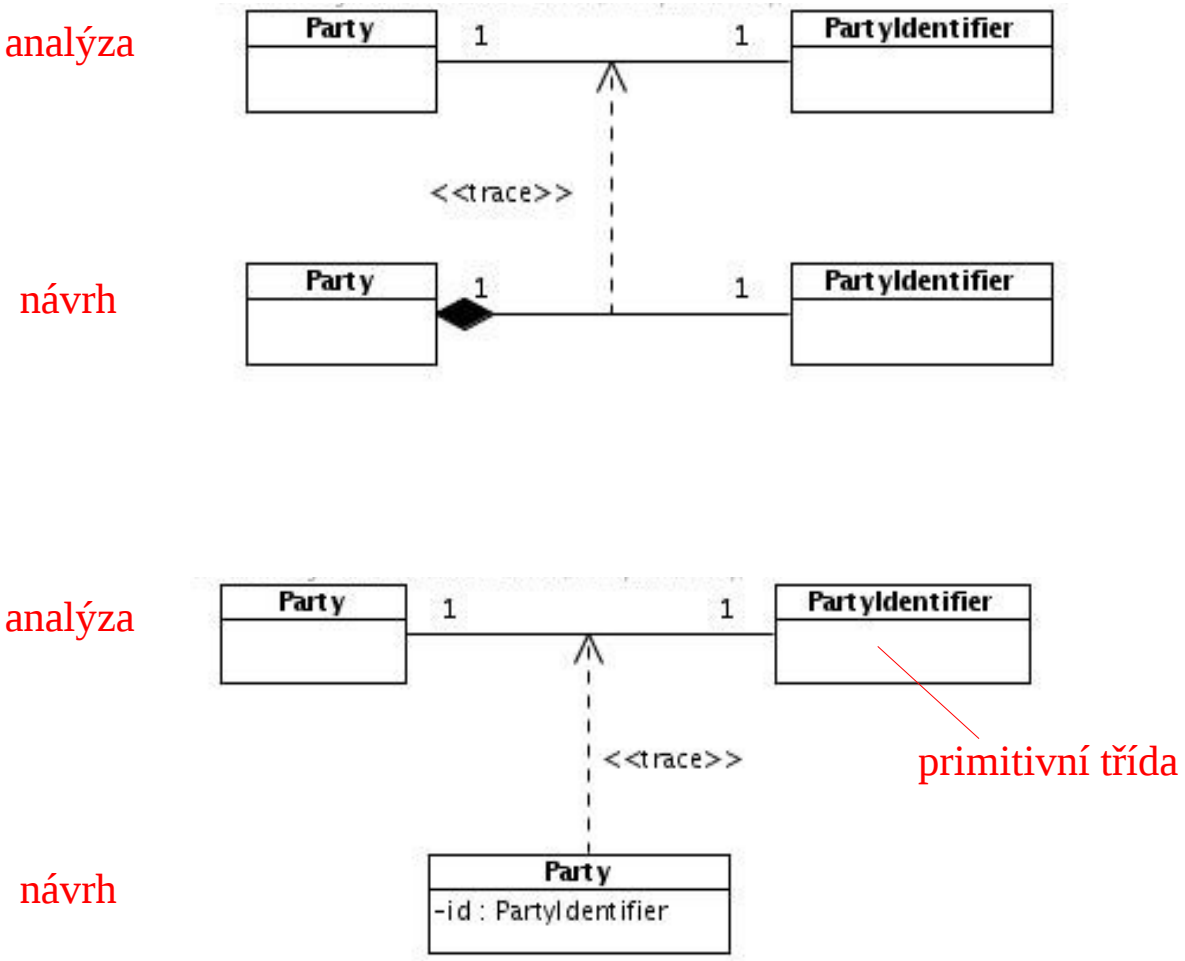
- I když je asociace jednosměrná, většinou je možné ji “procházet i v zakázaném směru” za cenu vyšší výpočetní náročnosti
- Př: Osoba neví nic o firmách, ve kterých je zaměstnána. Přesto můžeme firmy zaměstnávající osobu najít tak, že procházíme všechny firmy a hledáme ty, které zaměstnávají danou osobu.



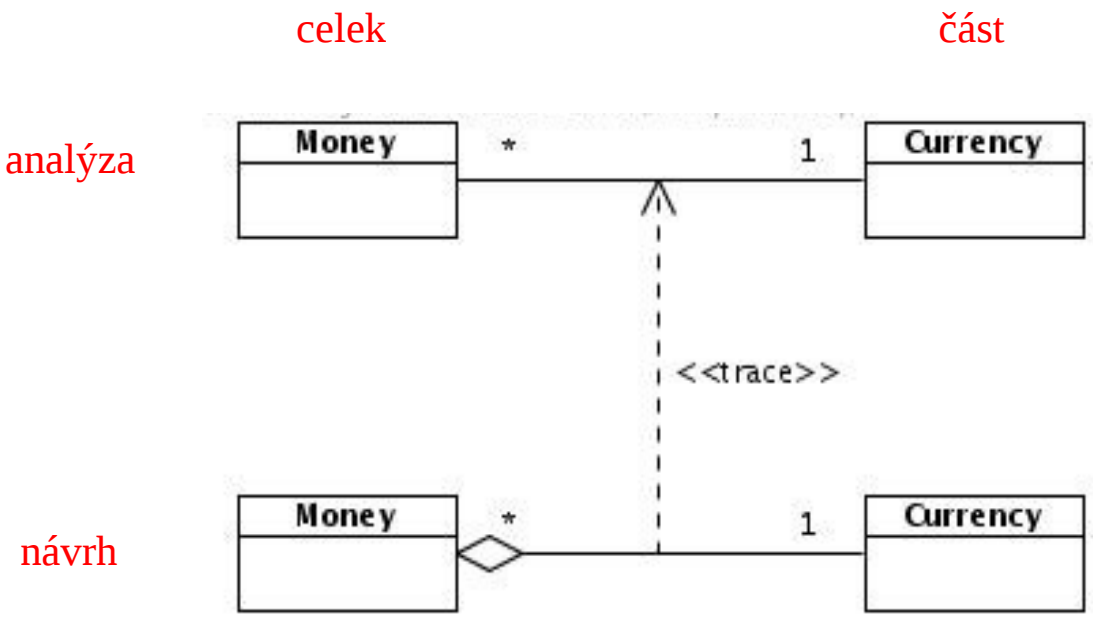
- Ekvivalent jednosměrných ulic: Snadno se dostaneme z jednoho konce na druhý. Pokud se chceme dostat zpět, musíme objet celý blok.
- V agregace/kompozici jsou většinou části řízeny svými celky



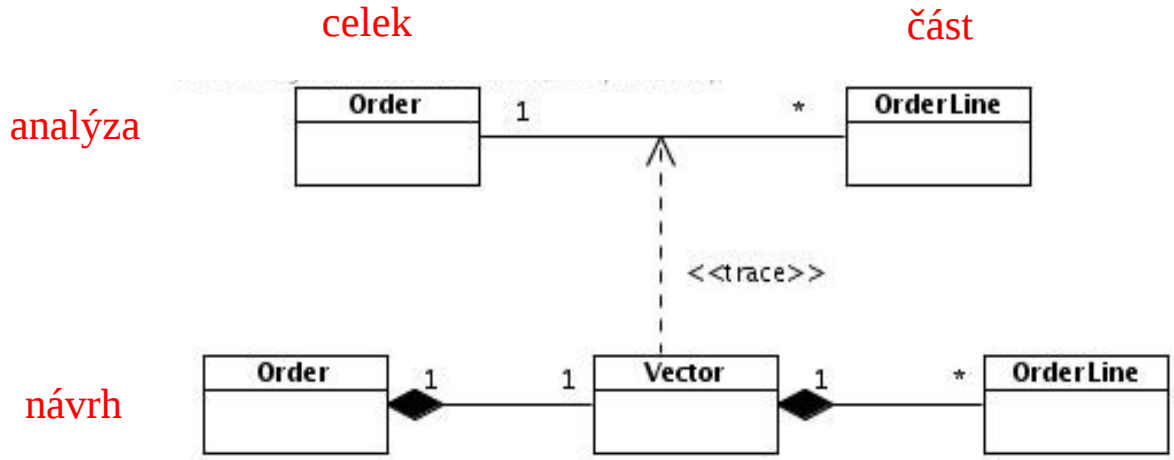
Návrhové třídy: Asociace 1:1

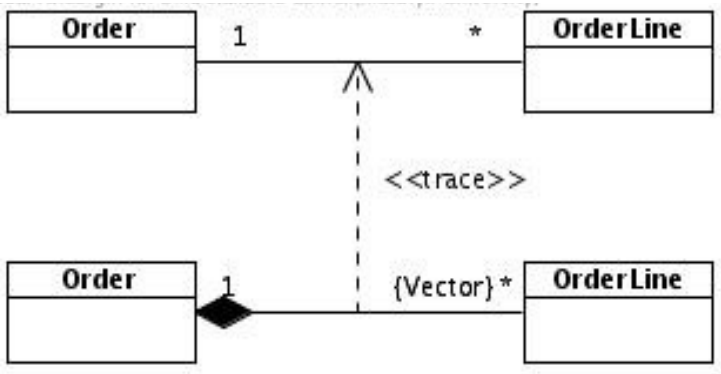


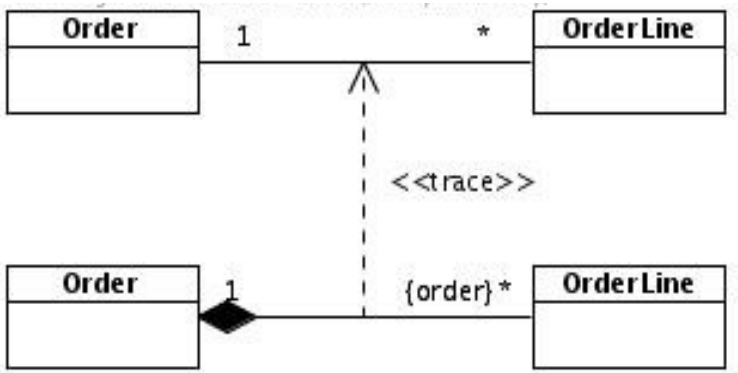
Návrhové třídy: Asociace M:1



Návrhové třídy: Asociace 1:M (kolekce)

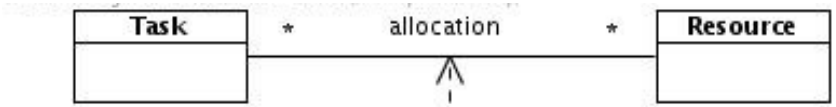


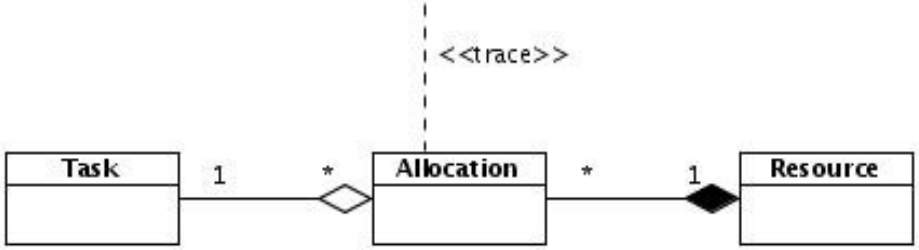




Návrhové třídy: Asociace M:N

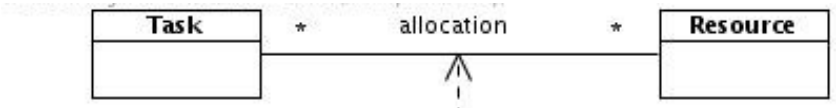
analýza

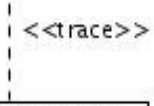




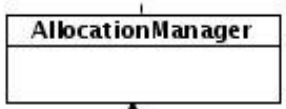
návrh

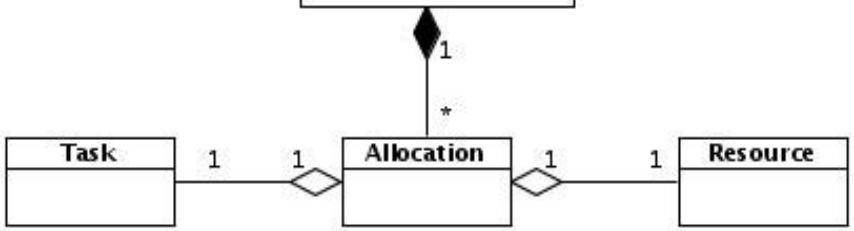
analýza





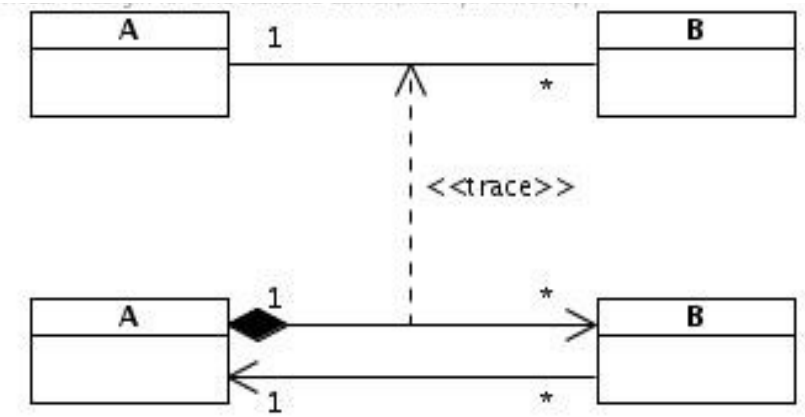
návrh





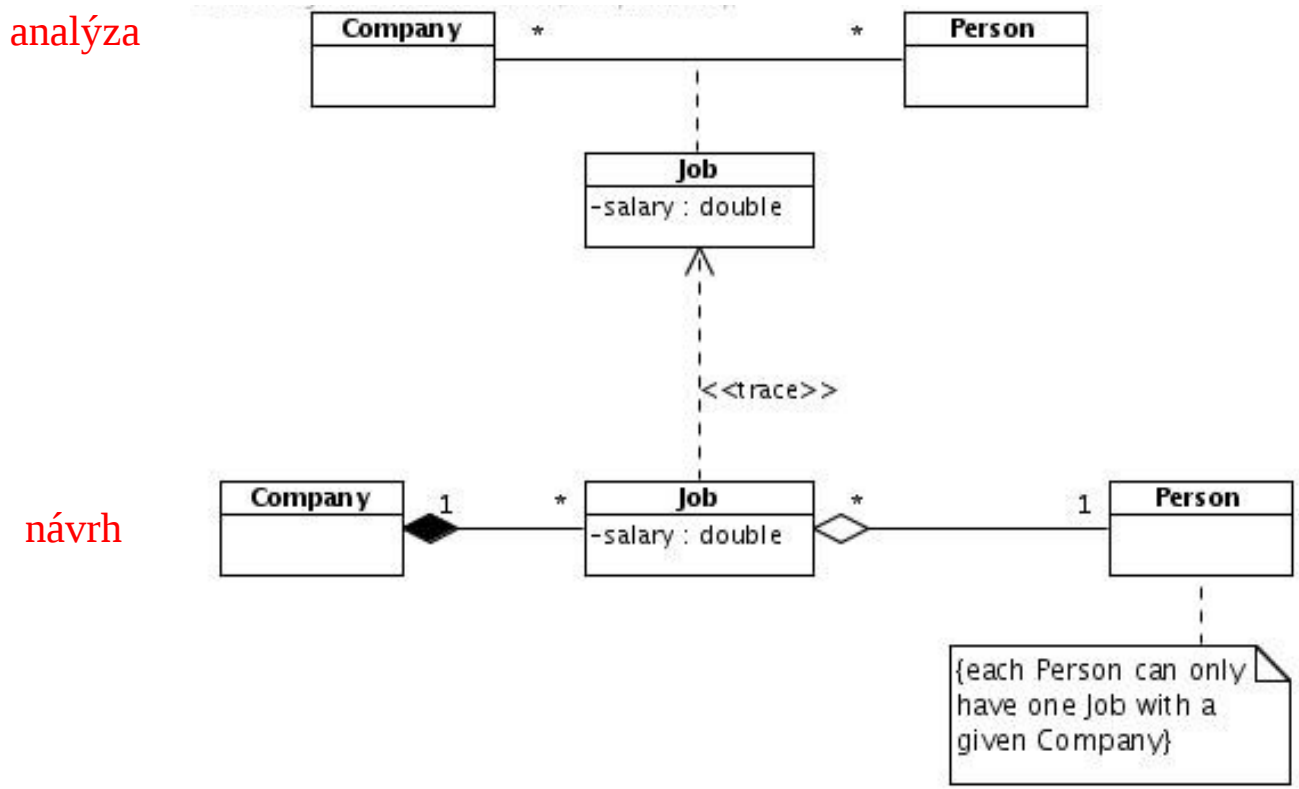
Návrhové třídy: Obousměrné asociace

analýza



návrh

Návrhové třídy: Asociační třídy



Návrhový model tříd

Příklad

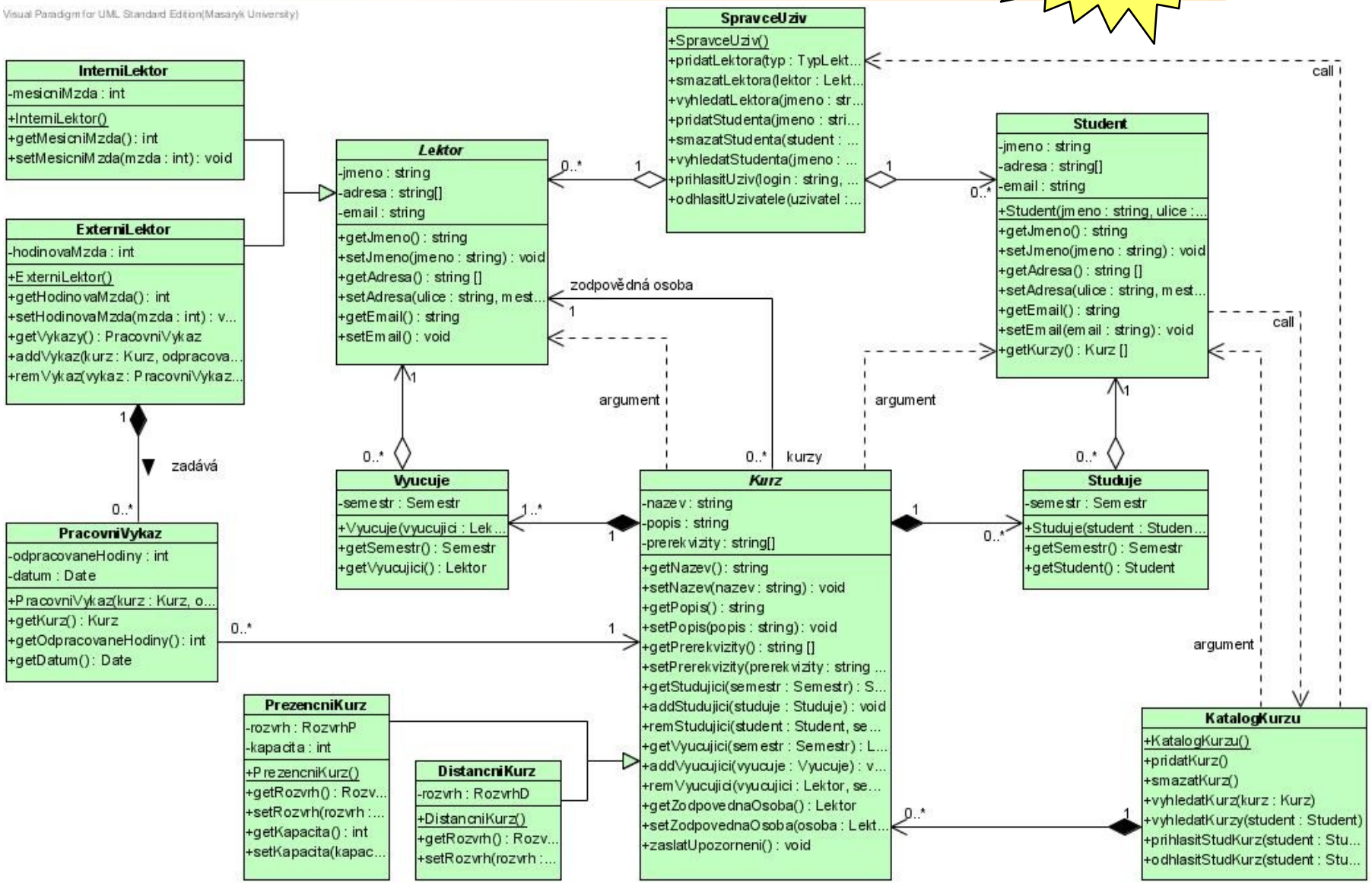
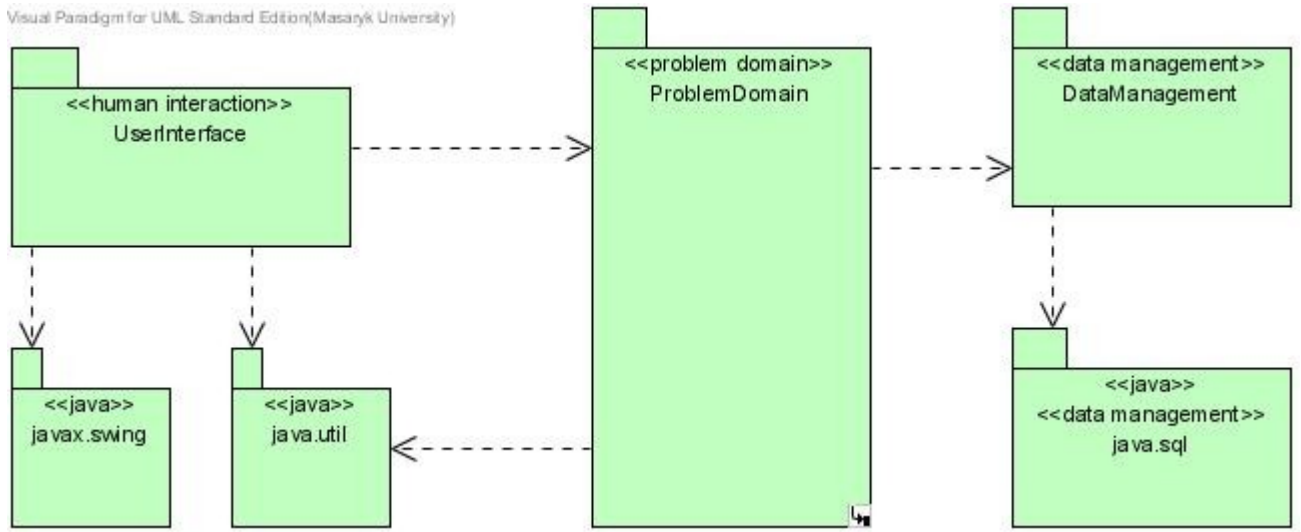
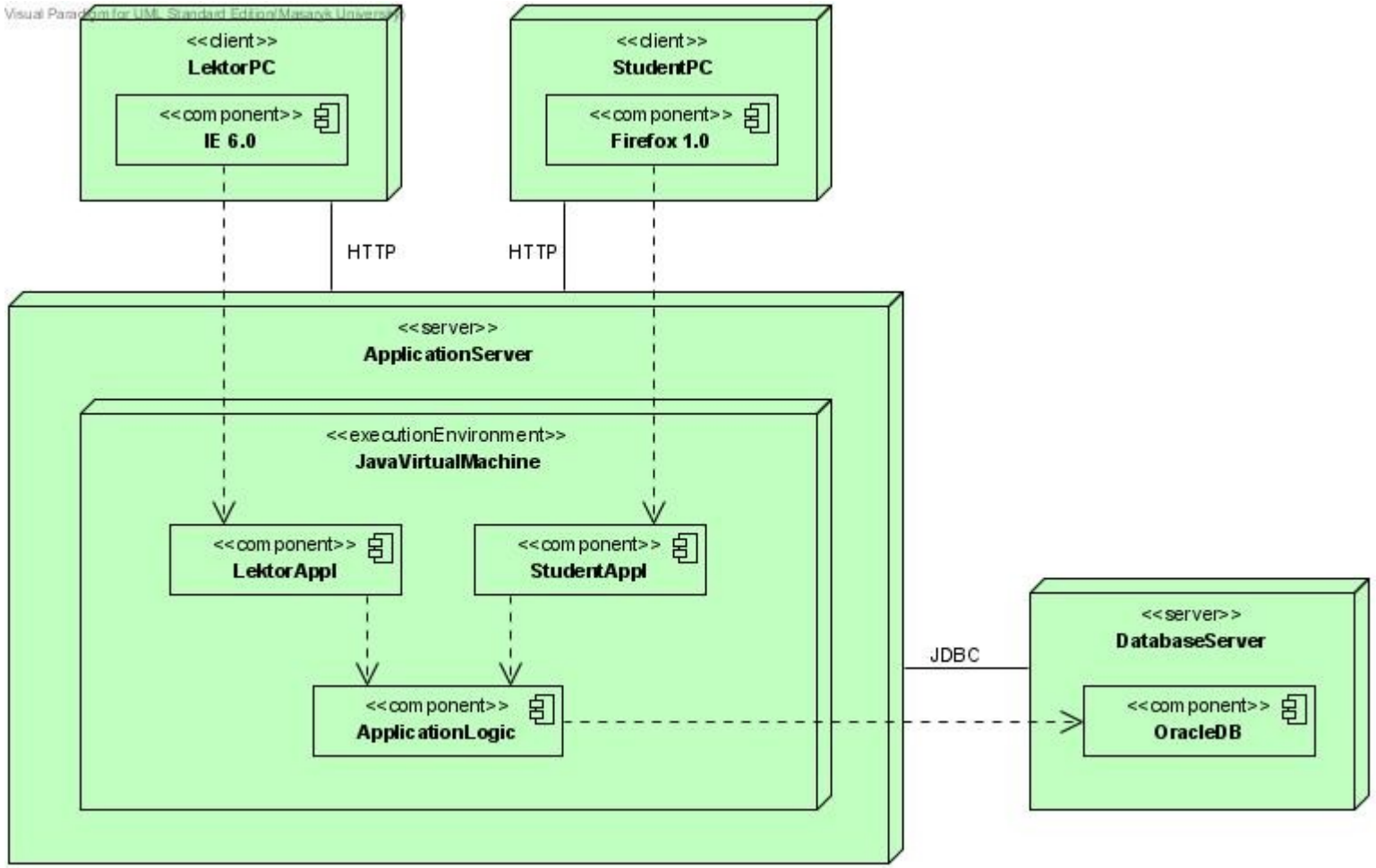


Diagram balíků

Příklad



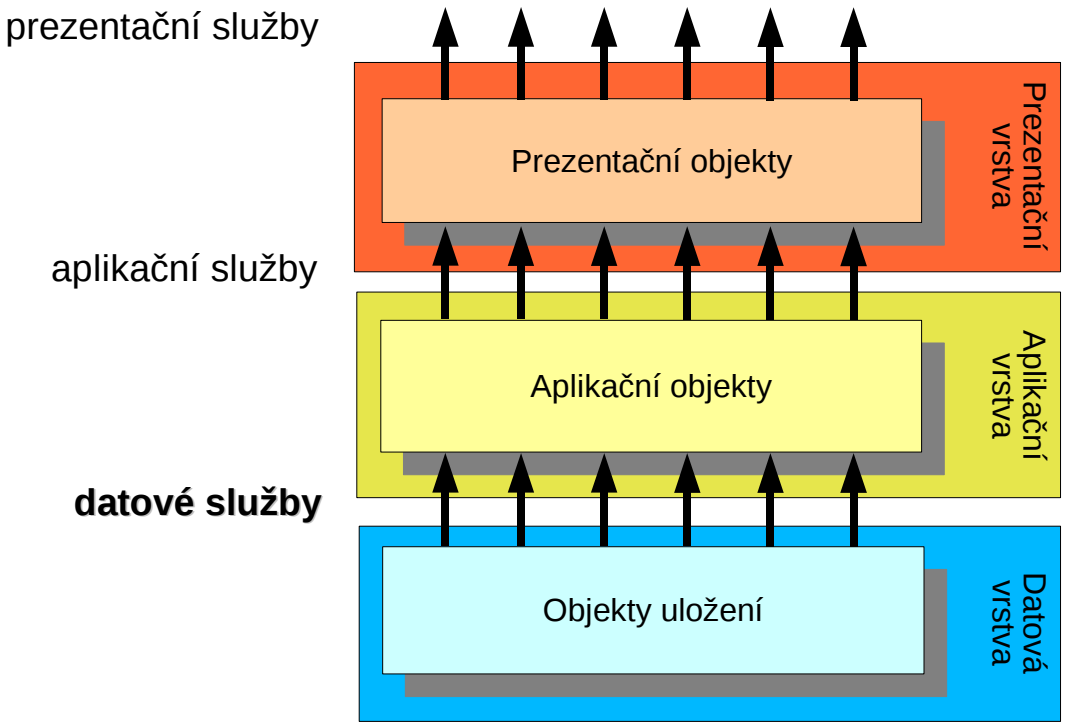
Příklad


Visual Paradigm for UML Standard Edition(Masaryk University)
<<client>>
LektorPC
<<component>>
IE 6.0
<<client>>
StudentPC
<<component>>
Firefox 1.0
HTTP
HTTP
<<server>>
ApplicationServer
<<executionEnvironment>>
JavaVirtualMachine
<<component>>
LektorAppl
<<component>>
StudentAppl
<<component>>
ApplicationLogic
JDBC
<<server>>
DatabaseServer
<<component>>
OracleDB

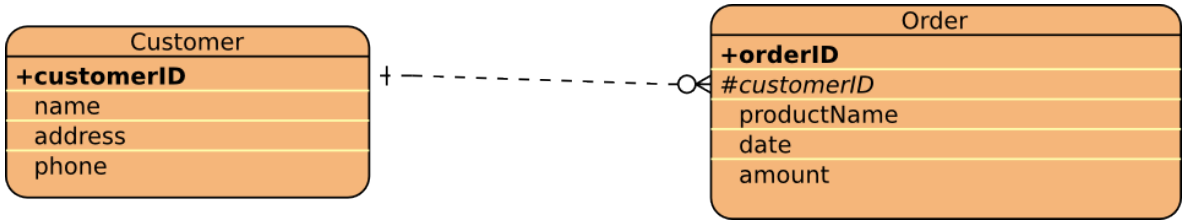
Návrh – uložení persistentních objektů

Proč potřebujeme model objektů uložení?

- Široké nasazení relačních databází, které ale nepodporují přímo struktury objektového modelu => nutnost mapování



Relační databáze

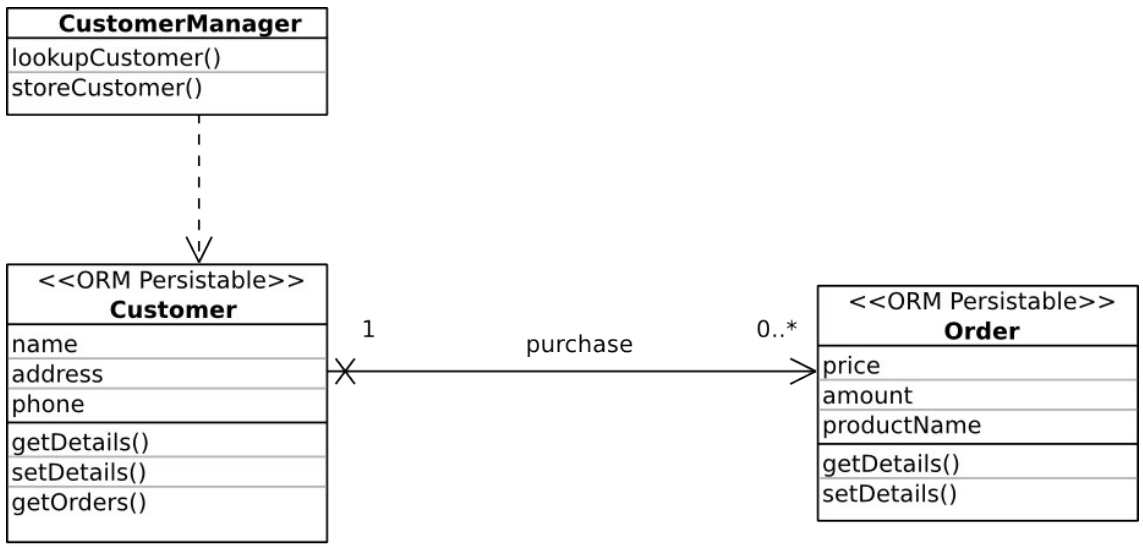


- Relační technologie
  - Uložení dat v tabulkách
  - Řádky jsou záznamy, sloupce typy
  - Tabulky jsou propojeny vazbami
  - Primární/cizí klíče
  - Kardinalita/násobnost vazeb
- Objektová technologie
  - Třídy obsahují data i operace
  - Asociace s násobností
  - Dědičnost

ORM – základní mapování

- Object-relational mapping, ORM
  - Persistentní třída definuje entitní množinu (tabulku)
  - Objekt definuje entitu (záznam, řádek tabulky)
  - Atributy třídy se stávají atributy entity (sloupce tabulky)
  - Klíč je vybrán z atributů nebo je vytvořen nový
  - Asociace/agregace/kompozice tříd definuje relaci (propojení tabulek cizími klíči)
  - Dědičnost tříd
    - mapování 1:1
    - zahrnutí do nadtřídy
    - rozpuštění do podtříd

ORM – základní mapování (II)

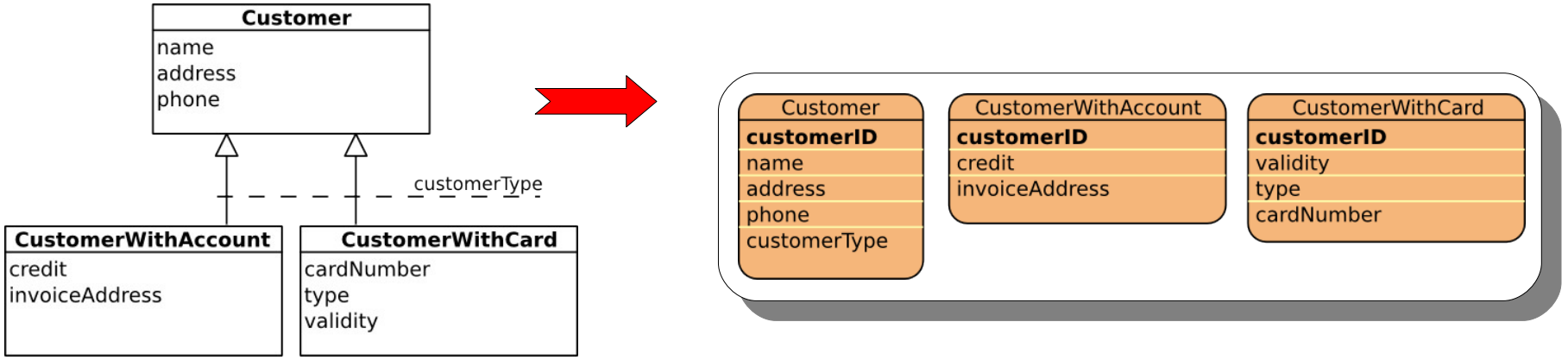


objekty pro práci s DB (SQL dotazy)

persistentní objekty

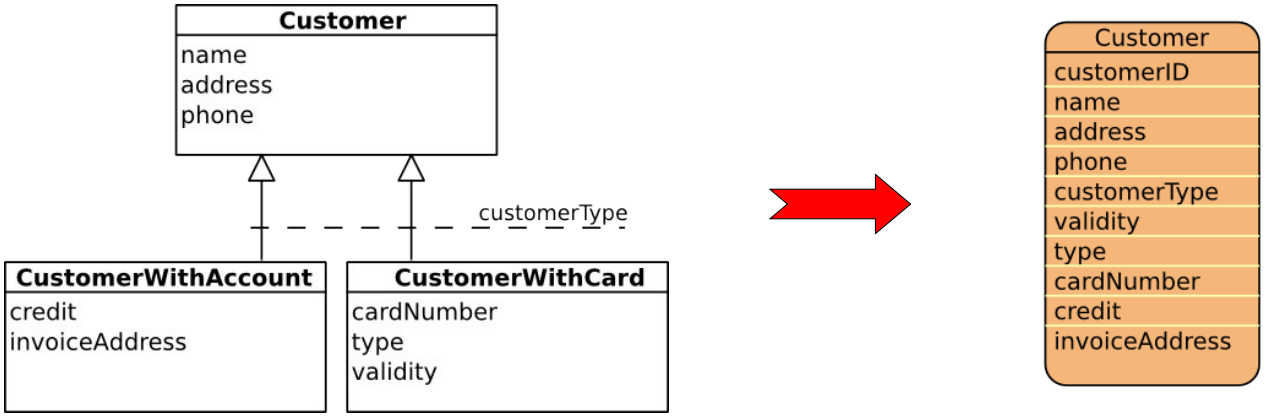
relační schéma

Dědičnost: mapování 1:1



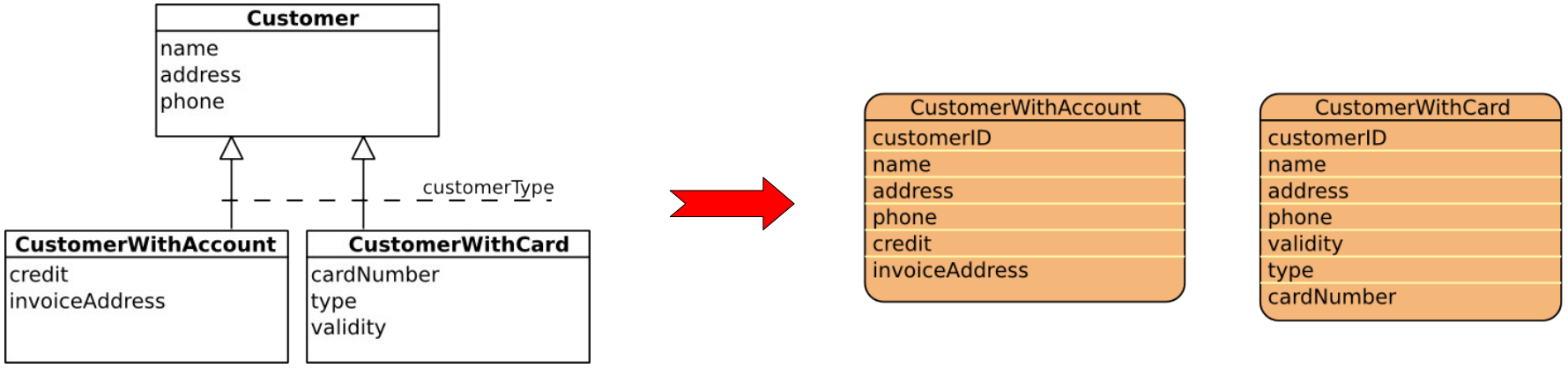
- Každá třída se stává tabulkou
- Všechny tabulky mají stejný primární klíč
- Diskriminátor se stává atributem
  - nutné pro ukládání podtříd přetypovaných na nadtřídu
- Instance obsahuje více tabulek
  - složitější přístup k datům

Dědičnost: zahrnutí do nadtřídy



- Všechny atributy podtříd jsou zahrnuty do jedné tabulky
- Některé atributy mohou obsahovat hodnotu NULL
- Vhodné v případě menšího počtu podtříd s málo atributy

Dědičnost: rozpuštění do podtříd



- Atributy nadtřídy jsou přeneseny do tabulek pro všechny neabstraktní podtřídy
- Vhodné při:
  - Nadtřída má málo atributů
  - Existuje mnoho podtříd (košatý strom větvení)
  - Podtřídy mají hodně atributů