PB007 Softwarové inženýrství I

Cvičenie 2 – organizácia testíkov, úvodný Use Case Diagram

Valdemar Švábenský

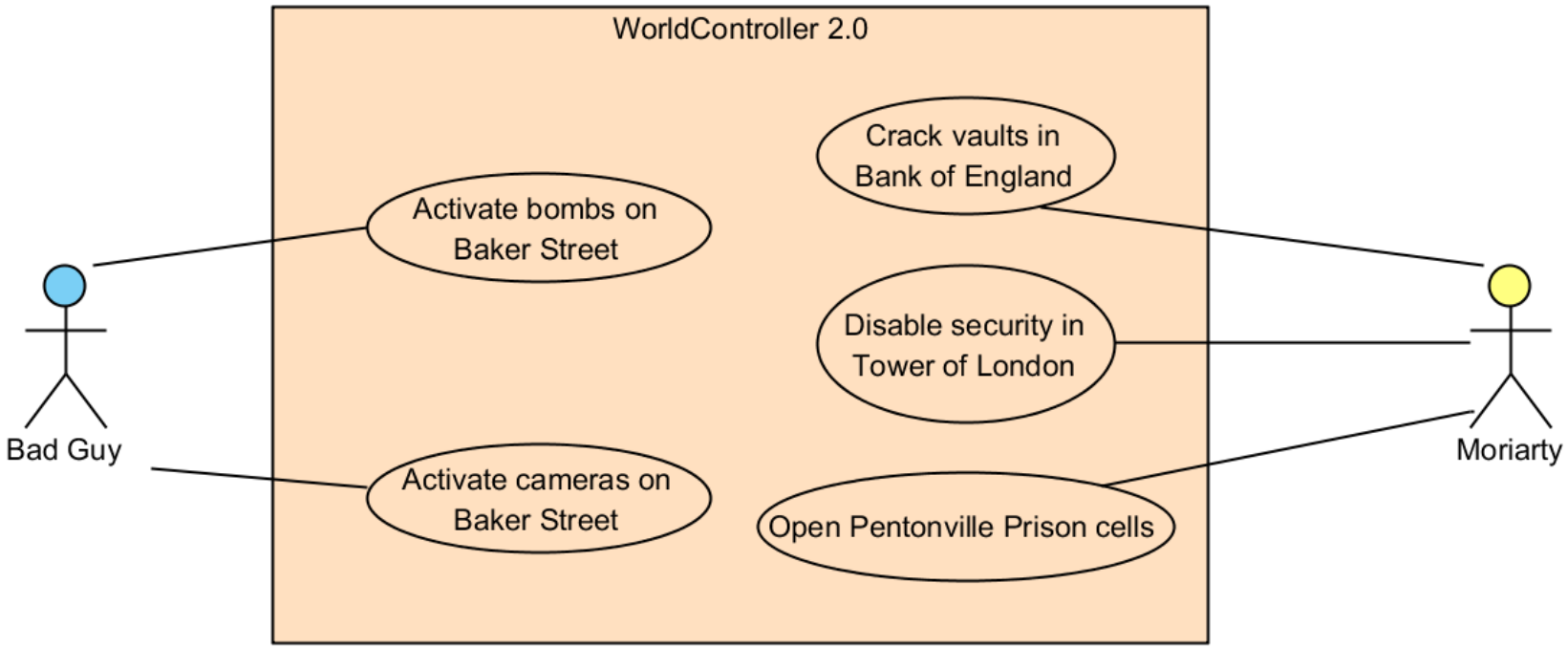
Fakulta informatiky, Masarykova univerzita, Brno

29. septembra 2015

Funkčné a nefunkčné požiadavky

- Požiadavky = popis systémových služieb a obmedzení
- Funkčné požiadavky = popis **služieb**, ktoré má systém poskytovať
  - Reakcie systému na isté situácie a vstupy
  - To, **čo** má systém robiť (alebo nerobiť)
  - Napr. „Systém umožňuje učiteľovi vytvoriť kurz“
- Nefunkčné požiadavky = popis vlastností systému a **obmedzení** na systém (spoľahlivosť, bezpečnosť, . . .)
  - **Ako** má systém niečo robiť
  - Napr. „Systém ukladá údaje o kurzoch v DBMS Oracle“
  - Buďte špecifickí

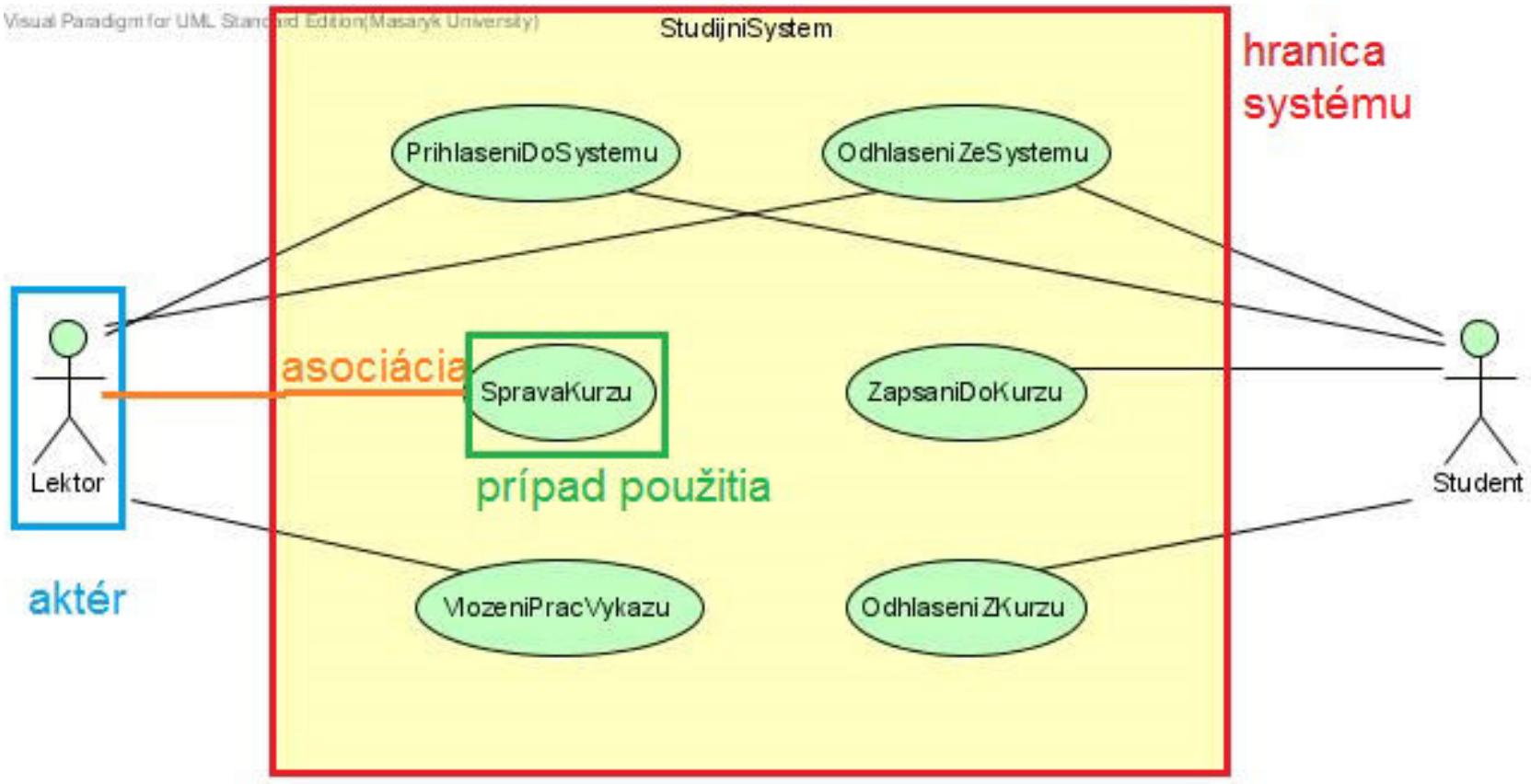
Use Case Diagram – ukážka (Sherlock)



Use Case Diagram

- Grafické vyjadrenie **funkčných požiadaviek** na systém
- Definuje **vonkajší pohľad** na systém
- Tvoria ho:
  - Hranica systému (System boundary / Subject)
  - Aktéri (Actors) = role entít externých k systému, ktoré priamo interagujú so systémom
  - Prípady použitia (Use cases) = funkcionality požadované (a spúšťané) aktérmi
  - Vzťahy / asociácie (Relationships) medzi aktérmi a prípadmi použitia
- Zachytáva komunikáciu medzi aktérmi a systémom

Use Case Diagram – ukážka



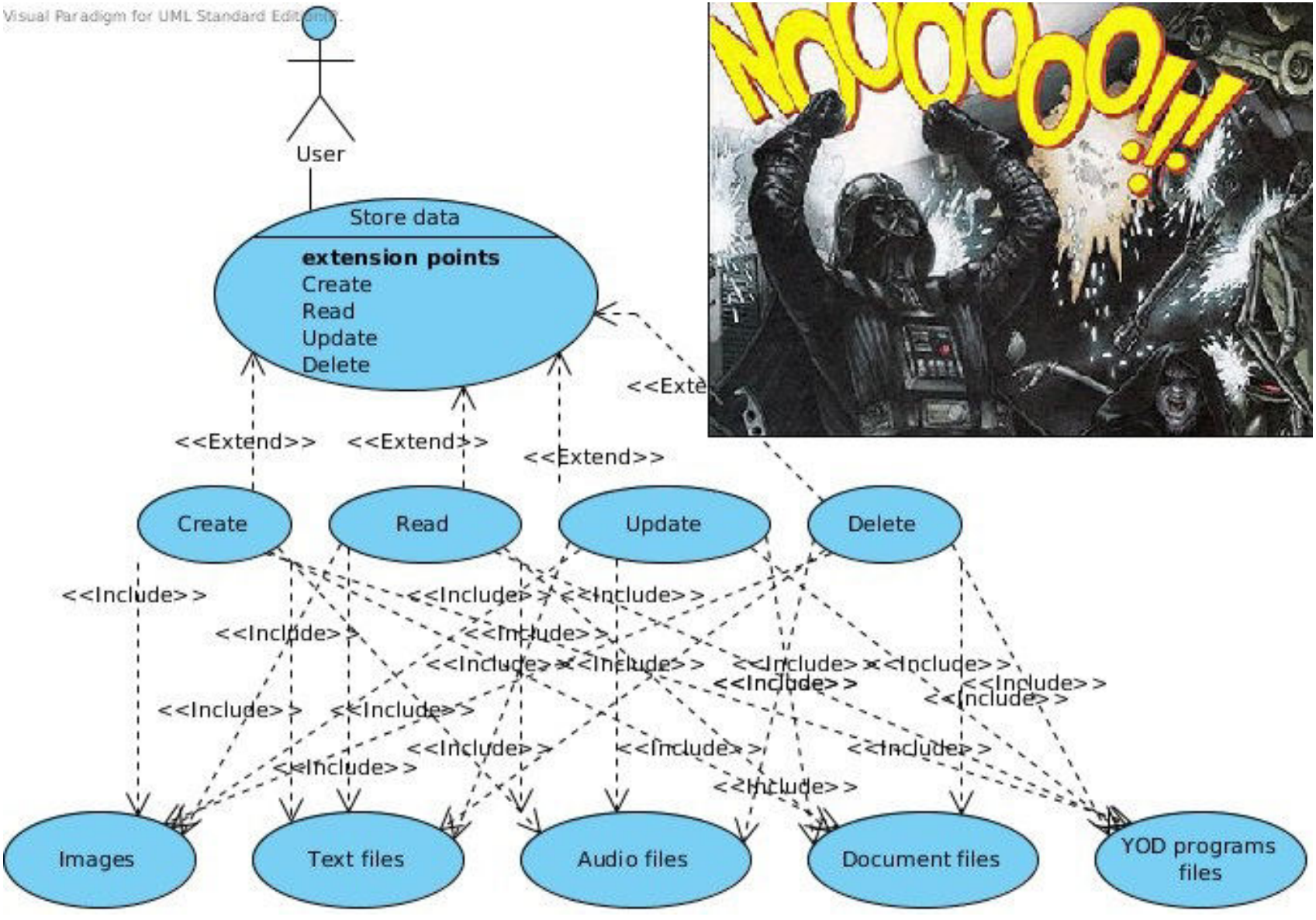
Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/02/02_Studium_UseCase.jpg

Postup pri modelovaní

1. Vymedzenie hraníc systému
2. **Nájdenie aktérov**
3. **Nájdenie prípadov použitia**
4. Určenie vzťahov medzi aktérmi a prípadmi použitia
5. Špecifikácia prípadov použitia

"Keep It Simple, Stupid"

Vytvorte prehľadný Use Case



Identifikácia aktérov

- Kto alebo čo používa daný systém?
- Akú rolu zohráva pri tejto interakcii?
- Aké ďalšie systémy spolupracujú s našim systémom?
- Kto alebo čo získava informácie zo systému?
- Kto alebo čo zadáva informácie do systému?
- Dochádza k nejakej udalosti pravidelne alebo v pevne danom čase?
- Používajte jasné a jedinečné označenie podstatným menom v jednotnom čísle

Identifikácia prípadov použitia

- Prípad použitia začína akciou tzv. **primárneho** aktéra
- Prípad použitia môže ovplyvniť tzv. **sekundárnych** aktérov
- Aké funkcie požaduje konkrétny aktér od systému?
- Ukladá a získava systém nejaké informácie? Ak áno, ktorí aktéri spúšťajú tieto činnosti?
- Čo sa stane pri zmene stavu systému? Sú o tom aktéri informovaní?
- Existujú externé udalosti, ktoré ovplyvňujú systém? Čo upozorní systém na tieto udalosti?
- Používajte jasné a jedinečné označenie v tvare slovesnej väzby z pohľadu aktéra

PB007 Softwarové inženýrství I

Cvičenie 3 – detailný Use Case Diagram

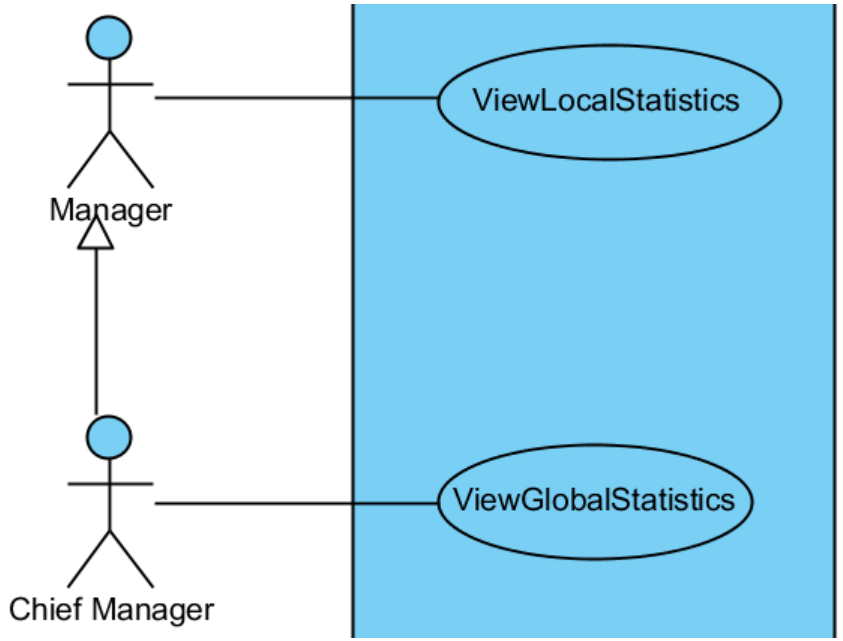
Valdemar Švábenský

Fakulta informatiky, Masarykova univerzita, Brno

6. októbra 2015

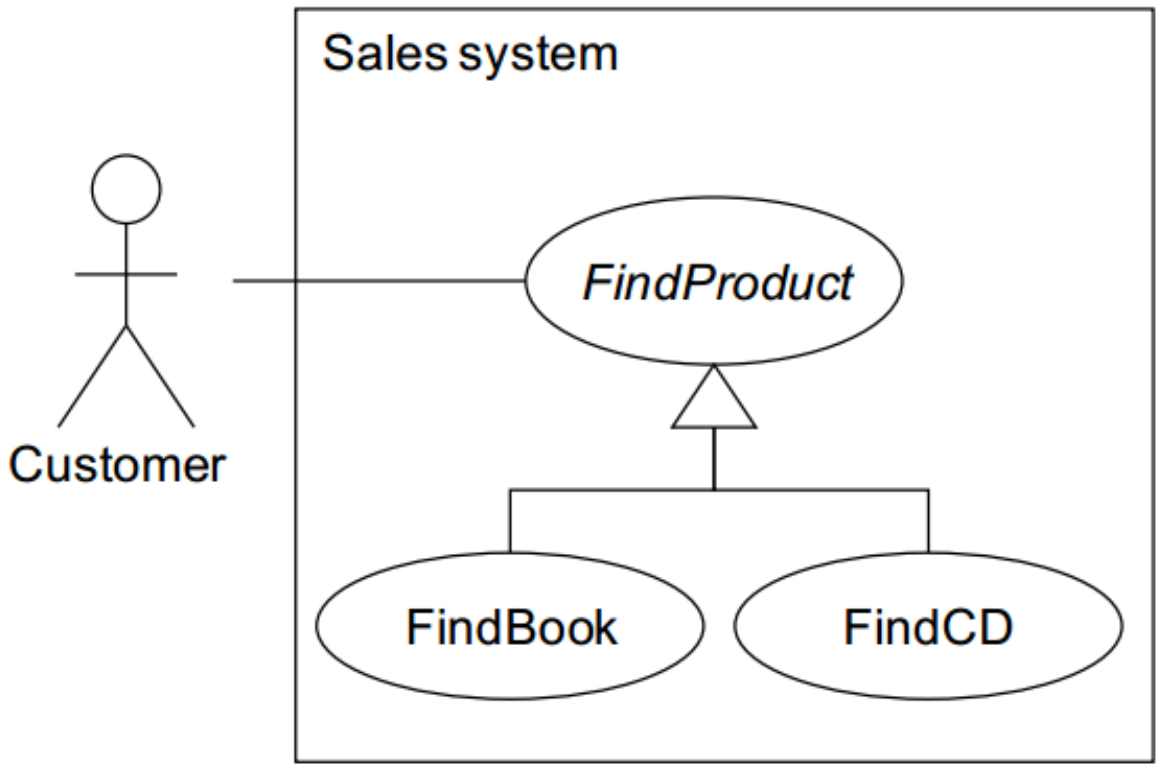
Generalizácia/špecializácia aktérov

- Vzťah všeobecného aktéra (predka) a špeciálneho aktéra (potomka)
- Potomok je **špecifickým prípadom** predka
- Môžeme vždy dosadiť potomka na miesto predka
- Potomok zdedí všetky väzby od predka
- Predok môže byť abstraktný
- Použite, keď to zjednoduší model



Generalizácia/špecializácia prípadov použitia

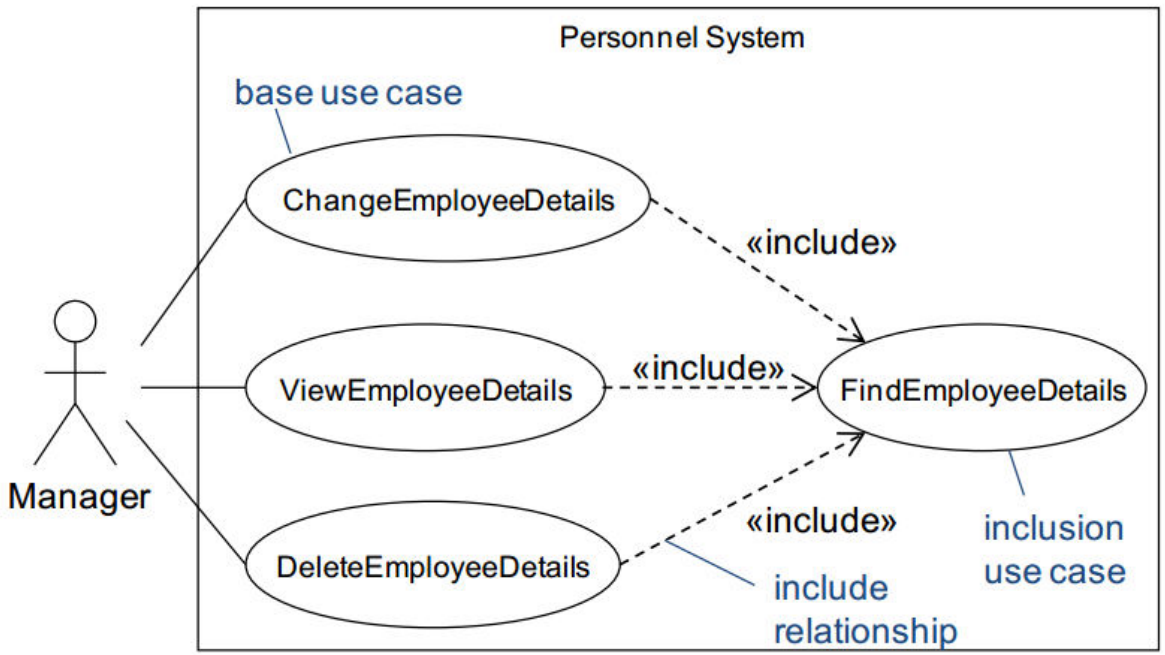
- Analogicky ako pri aktéroch
- Potomok môže pridať nové chovanie alebo pozmeniť chovanie predka
- Predok je často abstraktný



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/01-SoftwareDevelopment.pdf

Stereotyp «include» v prípadoch použitia

- Keď nejaké UC[1] **zdieľajú spoločné chovanie**, vynesieme toto spoločné chovanie do samostatného UC
- Takto vzniknutý UC „vložíme“ do základného UC
- Základný UC je neúplný bez vloženého UC
- Vkladaný UC môže a nemusí byť úplný sám o sebe

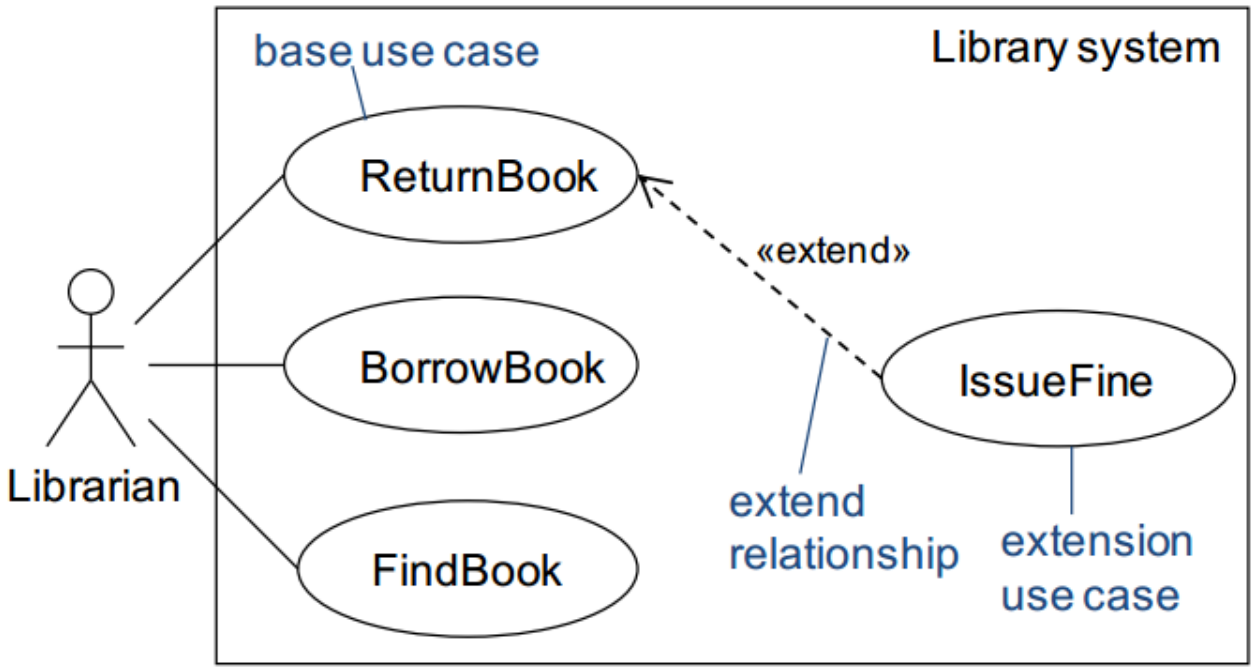


Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/01-SoftwareDevelopment.pdf

[1] Use Case(s)

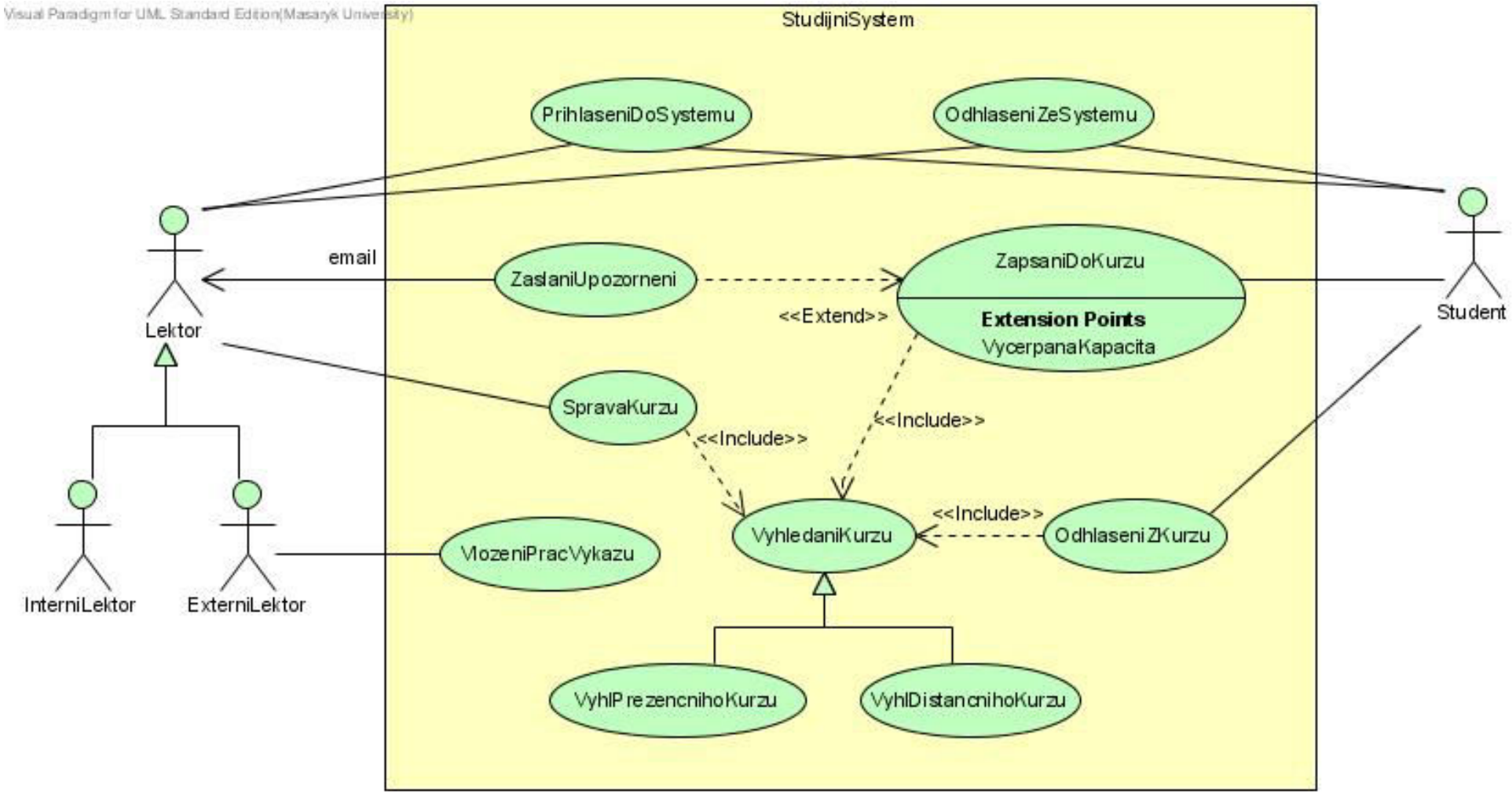
Stereotyp «extend» v prípadoch použitia

- Rozširujúci UC **za istých podmienok** pridáva akcie do základného UC
- Základný UC „nevie“ o rozširujúcom UC (rozšírenie sa niekedy ani nemusí vykonať)
- Základný UC je úplný aj bez rozširujúceho UC
- Rozširujúci UC môže a nemusí byť úplný sám o sebe



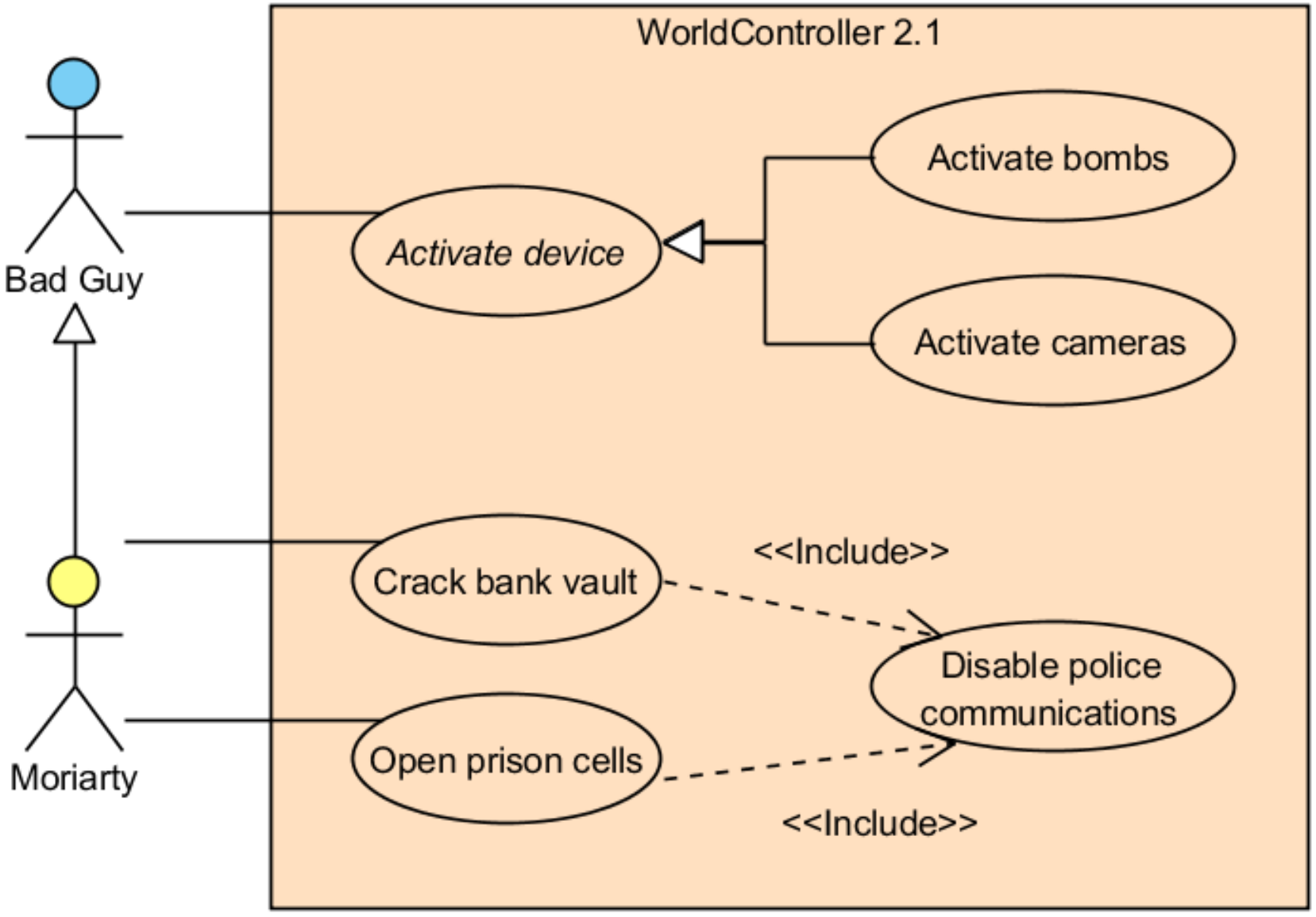
Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/01-SoftwareDevelopment.pdf

Use Case Diagram – ukážka



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/03/03_Studium_UseCase.jpg

Use Case Diagram – ukážka



Textová špecifikácia prípadov použitia

- Šablóna `StructuredFlow.udt` vo Visual Paradigm

ZapsaniDoKurzu

Main	
Use Case ID	1
Brief Description	UC1 umožní studentovi elektronický zápis do kurzu.
Primary Actors	Student
Secondary Actors	Lektor
Preconditions	Student je prihlasen do systemu
Main Flow of Events	1. Pripad uziti zacina, kdyz Studen zvoli v menu "Zapis do kurzu". 2. INCLUDE(VyhledaniKurzu) 3. POKUD byl vyhledan alespon jeden kurz 3.1. PRO KAZDY (vyhledany kurz, ktery nema doposud naplnenou kapacitu) 3.1.1. System zobrazi aktualni pocet prihlasenych studentu a nabidne volbu "Zapsat do kurzu". 3.2. POKUD Student zvoli "Zapsat do kurzu" 3.2.1. System zapise Studenta do kurzu. EXTENSION POINT(VycerpanaKapacita) 3.2.2. System potvrdi Studentovi uspesne prihlaseni do kurzu a aktualizuje zobrazeny pocet studentu v kurzu.
Post-conditions	Seznam studentu zapsanych do kurzu byl aktualizovan.
Alternative Flows	Student muze kdykoli opustit stranku pomoci volby "Zpet na uvodni stranku" nebo odhlasenim se ze systemu.

Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/03/03_Studium_1-ZapsaniDoKurzu.jpg

Textová špecifikácia prípadov použitia

- Name – názov presne ako v diagrame
- Use Case ID – jedinečný číselný identifikátor
- Brief Description – stručný popis
- Primary Actors – primárni aktéri
- Secondary Actors – sekundárni aktéri
- Preconditions – vstupné podmienky; musia byť splnené pred spustením prípadu použitia
- Main Flow of Events – hlavný tok udalostí; scenár
- Post-conditions – výstupné podmienky; musia byť splnené po dokončení prípadu použitia
- Alternative Flows – alternatívny tok udalostí vo výnimočných prípadoch (chyby, prerušenia, výnimky)

Hlavný tok udalostí („scenár“)

- Postupnosť jednotlivých krokov prípadu použitia
- Zobrazuje ideálny prípad (“happy day scenario”) bez výskytu chýb, prerušení a pod.
- Interakcia so systémom z pohľadu aktéra
- Krátke a jednoduché deklaratívne vety
- Číslované vety v časovej postupnosti
  - `<number> The <actor/system> <some action>`
  - `1. The use case starts when <actor> <action>`
- Je možné ho štruktúrovať
  - `IF`, `THEN`, `ELSE`, `FOR`, `WHILE`, `...` (verzálky kvôli rozlíšeniu od normálneho textu)
  - Odsadenie číslovania: 1. $\rightarrow$ 1.1.

Textová špecifikácia – tipy

- Doplňte všetky informácie potrebné k vykonaniu UC
  - Pozor na chýbajúce kroky, nedostatočné podmienky, ...
  - Špecifikujte dáta, ktoré požadujete, napr.: “User enters the name of the book and the author”
- Kontrolujte vstup, ktorý dostávate a upozornite používateľa na nesprávny vstup
- Píšte natoľko detailné popisy, aby ste ich chápali aj o rok, napr.: “System checks deadlines” vs. “System checks deadlines for returns of currently lent books”
- Namiesto spojky „a“ obvykle radšej rozpíšte dva kroky, chybou je napr.: “System saves the information in the database and notifies all users by e-mail”

PB007 Softwarové inženýrství I

Cvičenie 4 – Activity Diagram

Valdemar Švábenský

Fakulta informatiky, Masarykova univerzita, Brno

13. októbra 2015

Activity Diagram – základný popis

- Modeluje tok aktivity – postupnosti akcií
  - Aktivitou je napr. prípad použitia, metóda, algoritmus
- Môže dokumentovať zložité prípady použitia

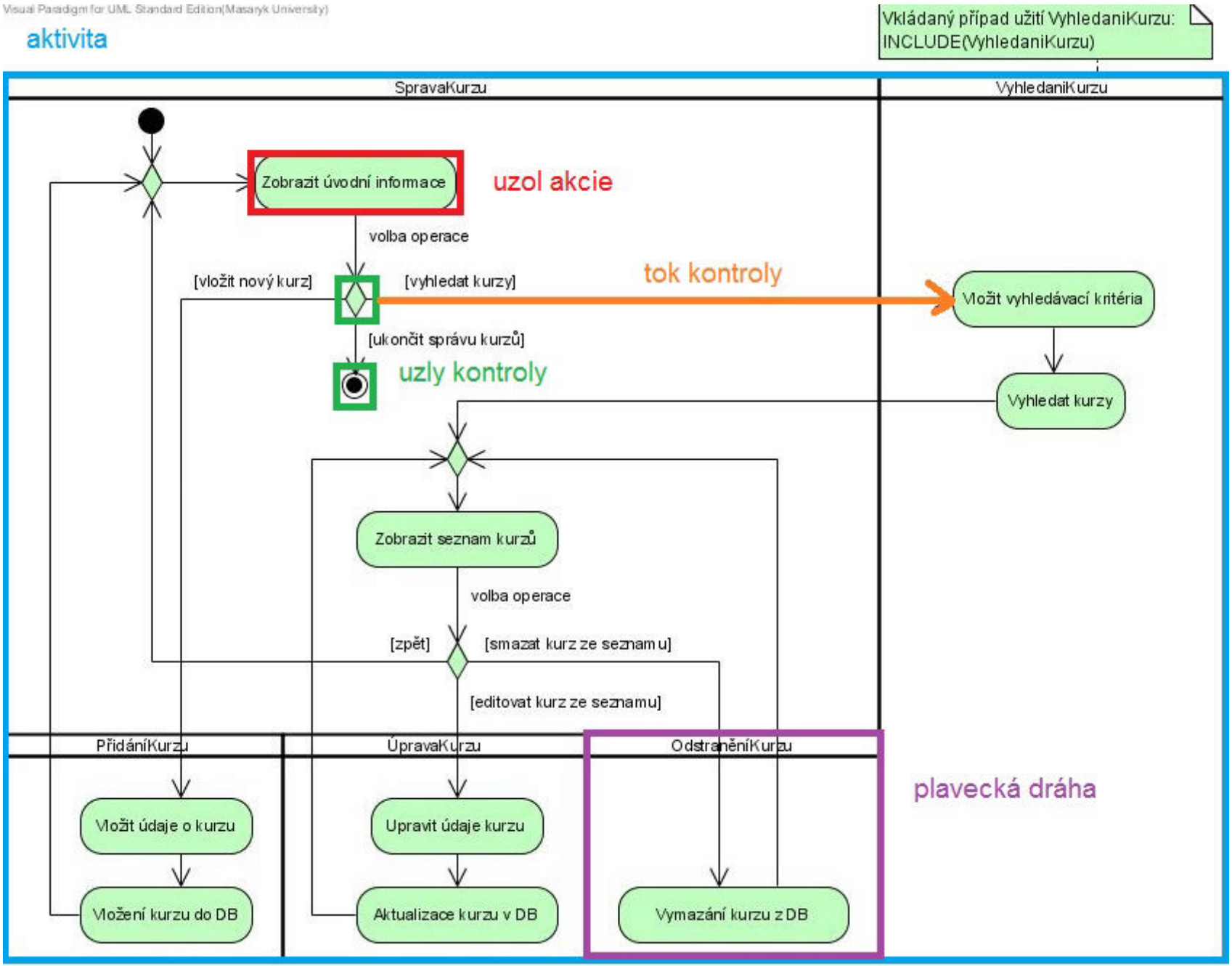
- Activity Diagram je objektový vývojový diagram
- Activity Diagram tvoria rôzne typy uzlov prepojené orientovanými hranami

Zdroje obrázkov: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/04/04_Studium_Activity_SpravaKurzu.jpg

https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/04/04_Studium_Activity_VlozeniUdaju.jpg

https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/02-RequirementsSpecification.pdf

Activity Diagram – ukážka

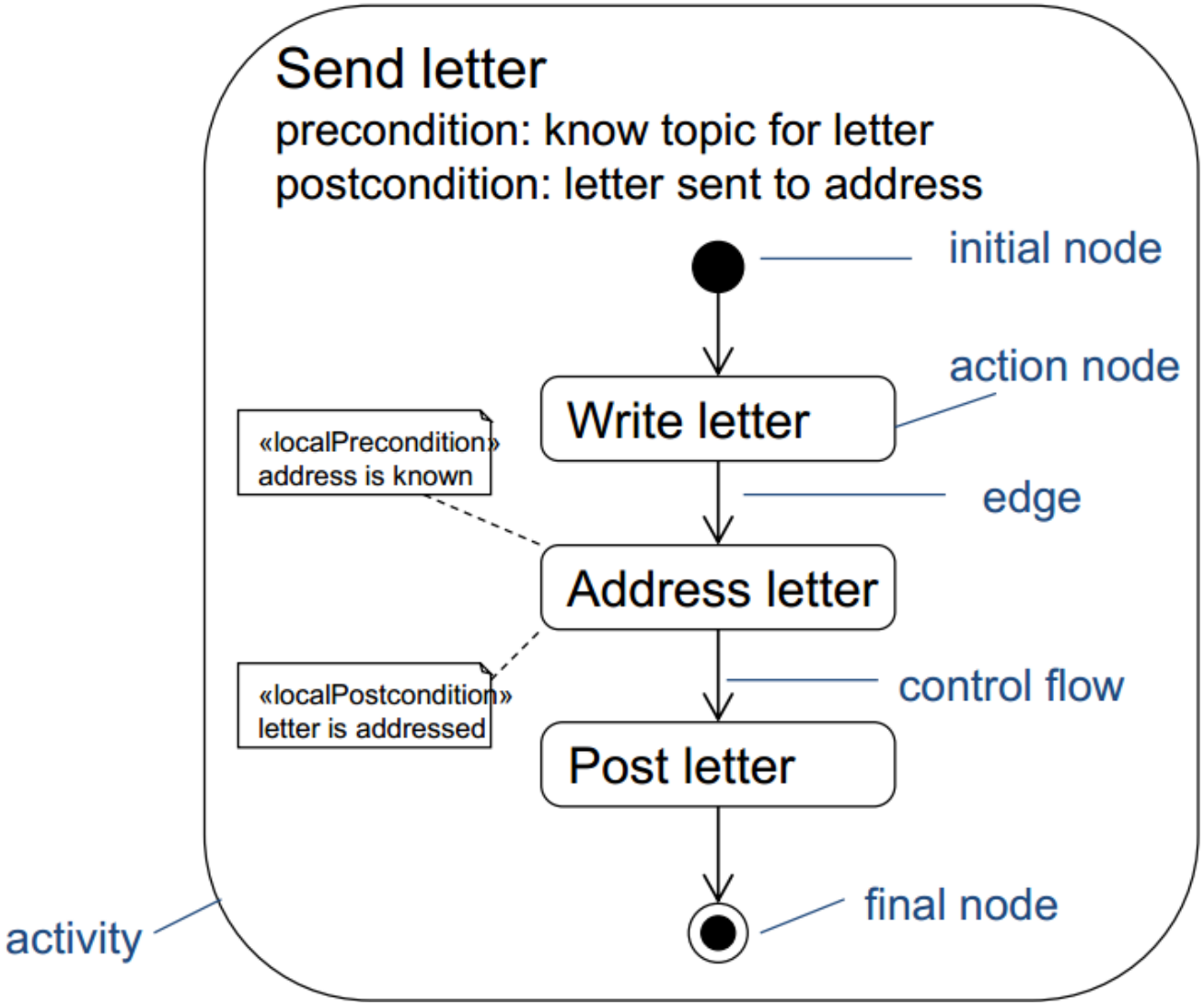


Activity Diagram – prvky

1. Token – imaginárny prvok, ktorý putuje po cestách v diagrame; reprezentuje napr. postup riadenia alebo dáta
2. Activity – aktivita, činnosť, ktorú je možné dekomponovať na sieť uzlov a tokov
3. Nodes – uzly
   - Action Nodes – nedeliteľné, neprerušiteľné, atomické akcie
   - Control Nodes – riadia cestu v rámci aktivity
   - Object Nodes – zastupujú objekty v rámci aktivity
4. Flows – toky (hrany)
   - Control Flow – riadiaci tok, cesta medzi uzlami
   - Object Flow – objektový tok, cesta medzi objektmi
5. Swimlanes – plavecké dráhy; logické zoskupenie súvisiacich akcií / tried / organizačných jednotiek

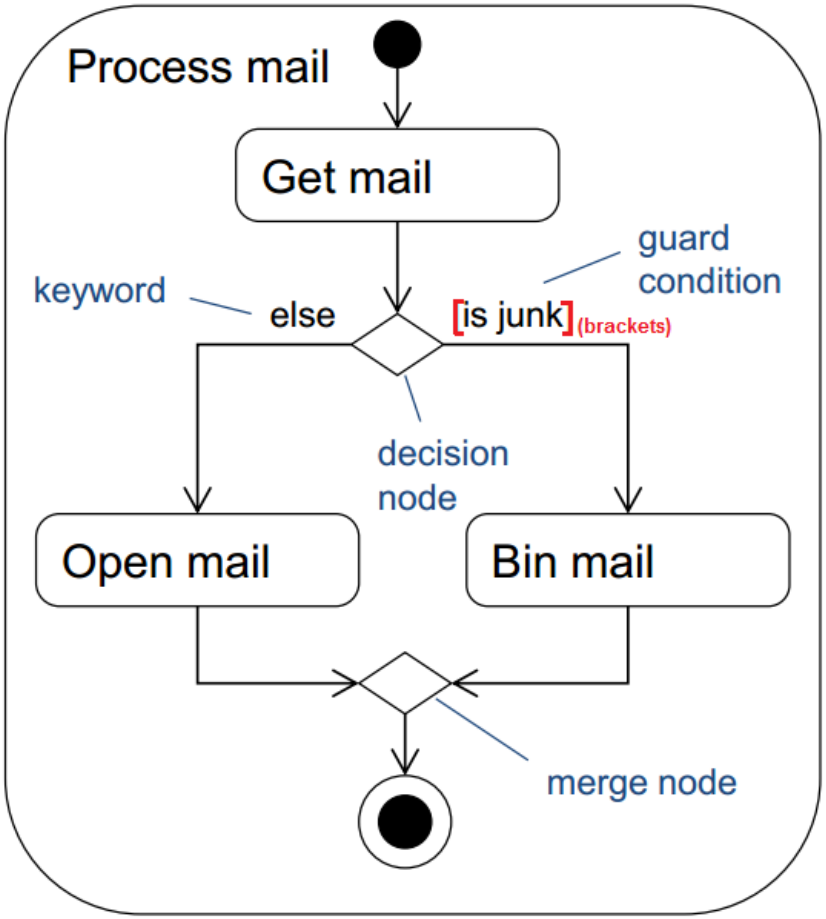
Riadiace uzly – počiatok a koniec

- Initial Node – počiatok toku po spustení aktivity
- Final Node – koniec aktivity



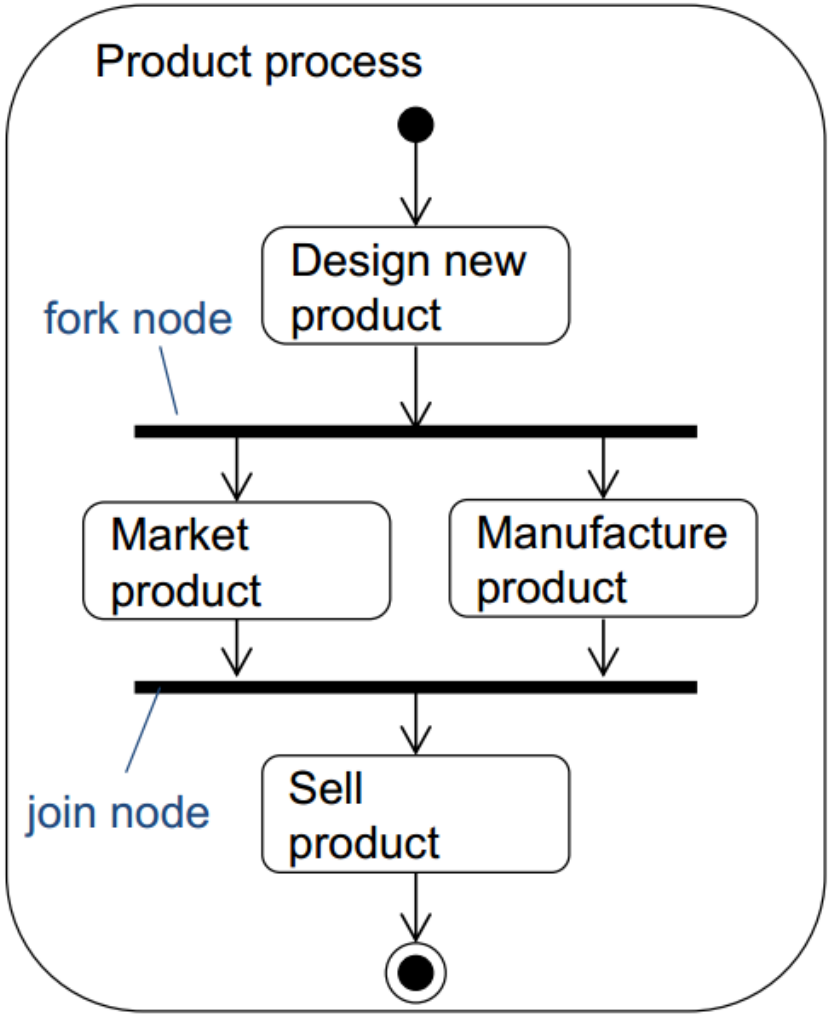
Riadiace uzly – podmienené vetvenie

- Decision Node – uzol rozhodovania (vetvenia); token pokračuje výstupnou hranou, ktorá splnila podmienku
- Merge Node – **zlučuje toky rozdelené vetvením**; skopíruje vstupné tokeny na výstupnú hranu

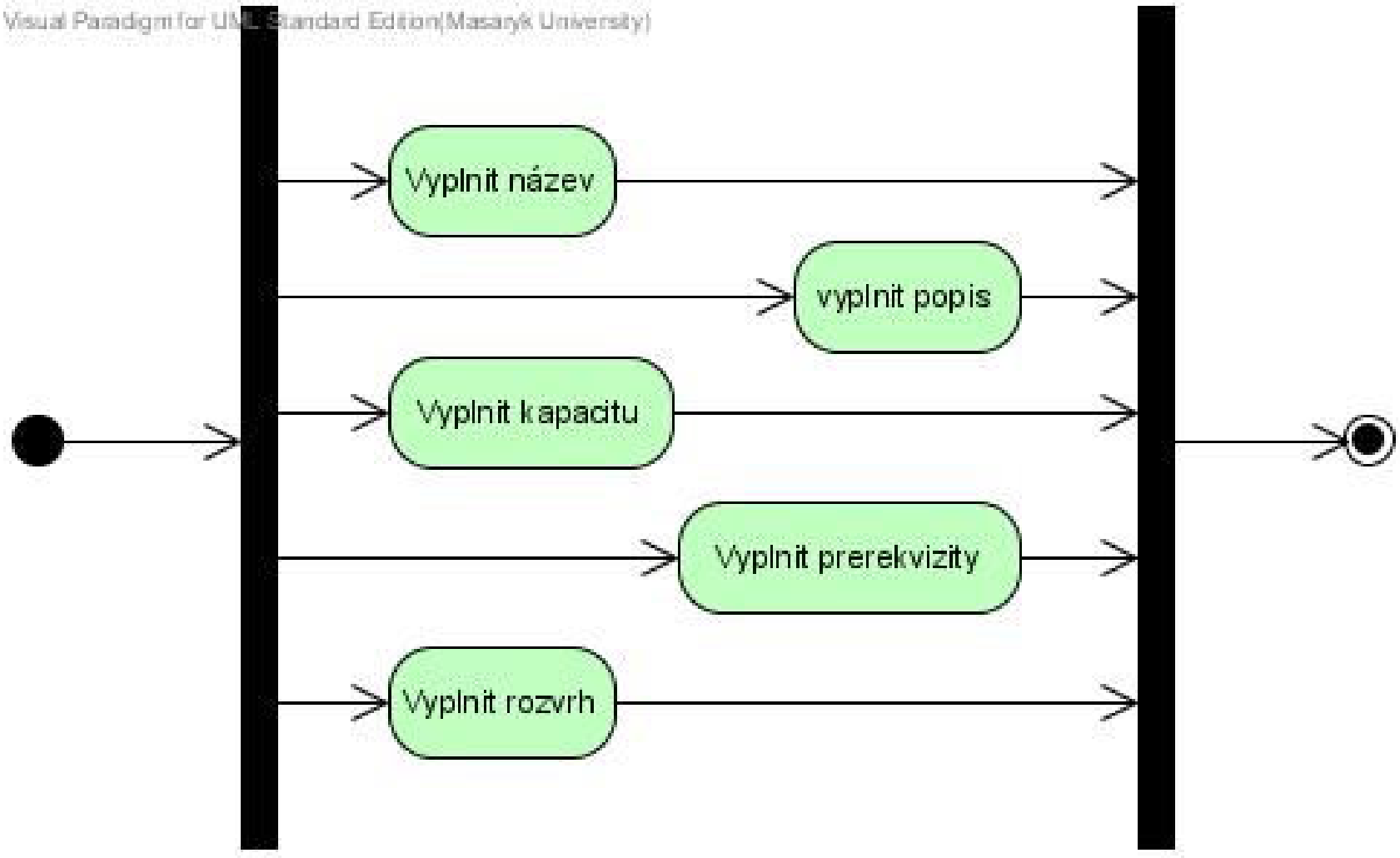


Riadiace uzly – súbežnosť

- Fork Node – rozdeľuje cestu na niekoľko súbežných tokov
- Join Node – spája a synchronizuje súbežné toky; čaká, kým budú tokeny na všetkých vstupných hranách



Activity Diagram – ukážka (paralelný beh aktivít)



Activity Diagram – relácie «include» a «extend»

«include»:

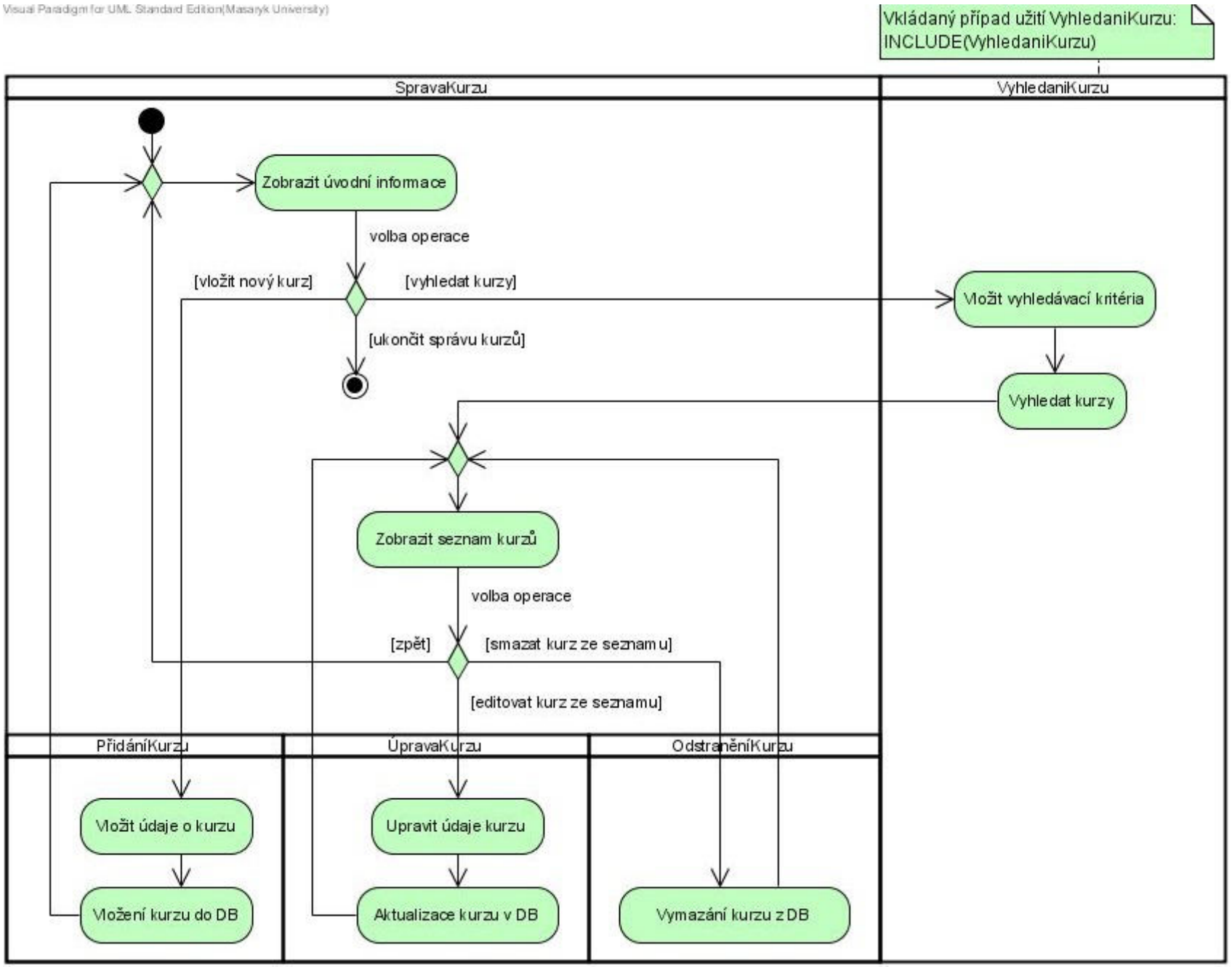
- Aktivity základného a vloženého UC[1] sú oddelené – majú svoje vlastné plavecké zóny

«extend»:

- Ak má byť bod rozšírenia vložený medzi *Akcia1* a *Akcia2*, za *Akcia1* sa vloží rozhodovací uzol pre vetvenie
  - Jeden prechod bude smerovať do *Akcia2*
  - Druhý prechod bude smerovať do rozširujúceho UC
- Po ukončení rozširujúceho UC sa opäť prejde do *Akcia2*
- Každý z UC môže byť vo vlastnej plaveckej zóne

[1] Use Case(s)

Activity Diagram – ukážka (plavecké dráhy)



Activity Diagram – tipy

- Začnite tvorbou (vertikálnych) plaveckých dráh (max. 5)
- Umiestnite počiatočný uzol do ľavého horného rohu
  - Číta sa zľava doprava a zhora dole
- Akčné uzly by mali mať vstupné a výstupné hrany
  - Ak nemajú vstupné hrany, je to to isté, ako keby boli spojené s počiatočným uzlom
- Akčné uzly majú obvykle 1 vstupnú a 1 výstupnú hranu
  - Viacero v/v šípok znamená paralelný beh akcií
  - Nutnosť Merge Node pri výstupe vetvenia
- Akčné uzly sú popísané slovesnou väzbou
- Výstupné podmienky z rozhodovacieho uzlu sú disjunktné (majú prázdny prienik) a pokrývajú všetky možnosti
  - Pozor na ∞ cykly pri opakovanom (ne)splnení podmienky

PB007 Softwarové inženýrství I

Cvičenia 5 a 6 – Analytic Class Diagram

Valdemar Švábenský

Fakulta informatiky, Masarykova univerzita, Brno

20. októbra 2015

Class Diagram – úvod

- Diagram tried je **najčastejšie používaný diagram** pri modelovaní objektového systému
- Modeluje statickú štruktúru systému: množinu tried, rozhraní a vzťahy medzi triedami
- Je rozsiahly, preto sa jeho tvorba delí do viacerých krokov:

1. Slovná analýza zadania a hľadanie tried
2. Tvorba analytického diagramu tried (Analytic CD)
   - Stručný, bez implementačných detailov
3. Tvorba návrhového diagramu tried (Design CD)
   - Vzniká pridaním implementačných detailov do analytického diagramu tried

Príklad výstupu kroku 1 (slovná analýza)

Výsledné třídy, atributy a operace po doplnění v podobě CRC:

Třída: Kurz/PrezencniKurz/DistancniKurz
Atributy: nazev, popis, prerekvizity, kapacita, rozvrh
Operace (zodpovědnosti): pridatKurz, smazatKurz, upravitUdaje, prihlasitStudenta, odhlasitStudenta, zaslatUpozorneni
Spolupracovníci: Studenti, zodpovědný Lektor, vyučující Lektoři

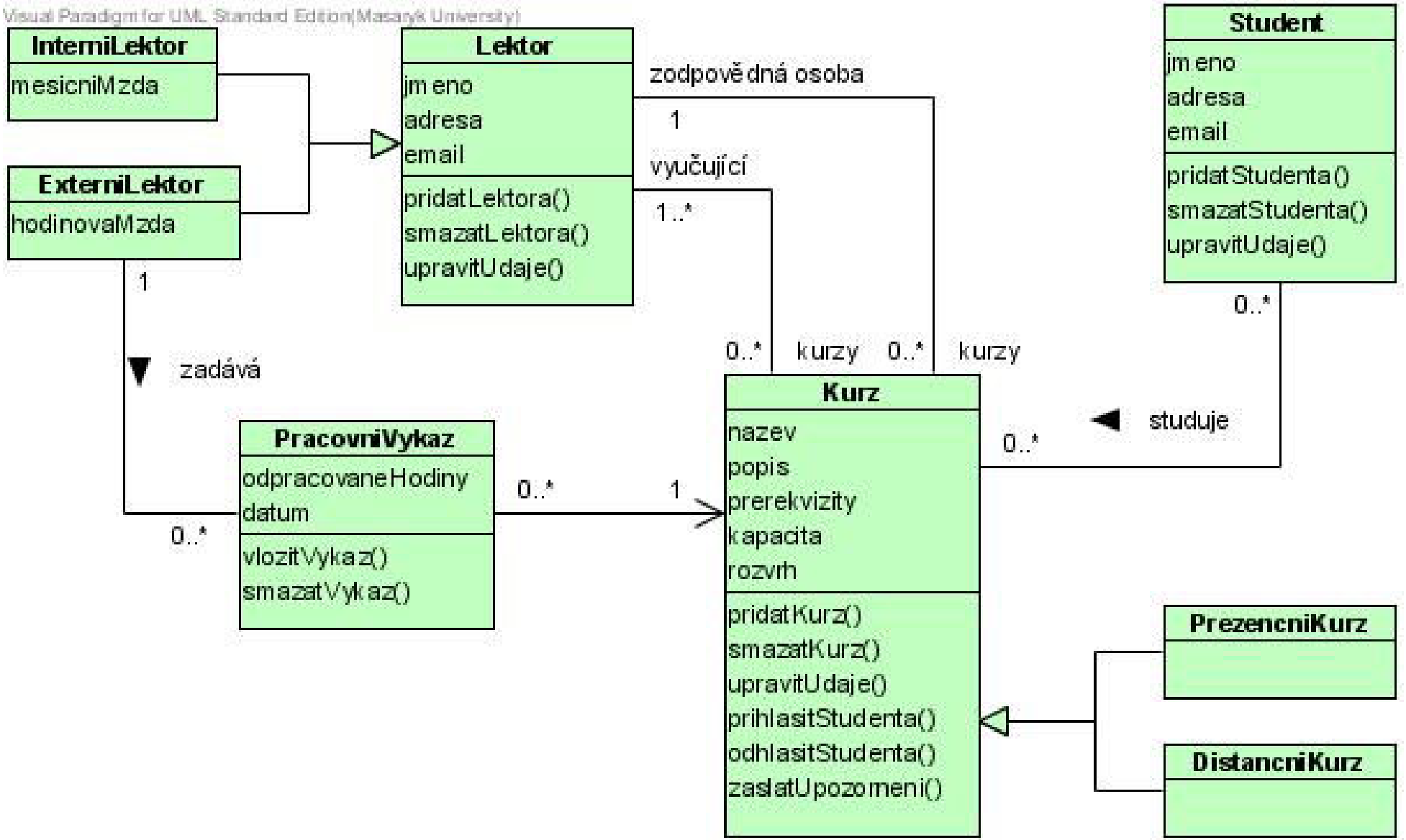
Třída: Student
Atributy: jmeno, adresa, email
Operace (zodpovědnosti): pridatStudenta, smazatStudenta, upravitUdaje
Spolupracovníci: Kurzy

Třída: Lektor/InterniLektor/ExterniLektor
Atributy: jmeno, adresa, email, mesicniMzda/hodinovaMzda
Operace (zodpovědnosti): pridatLektora, smazatLektora, upravitUdaje
Spolupracovníci: PracovniVykaz, vyučované Kurzy, spravované Kurzy

Třída: PracovniVykaz
Atributy: odpracovaneHodiny, datum
Operace (zodpovědnosti): vlozitVykaz, smazatVykaz
Spolupracovníci: Lektor, Kurz

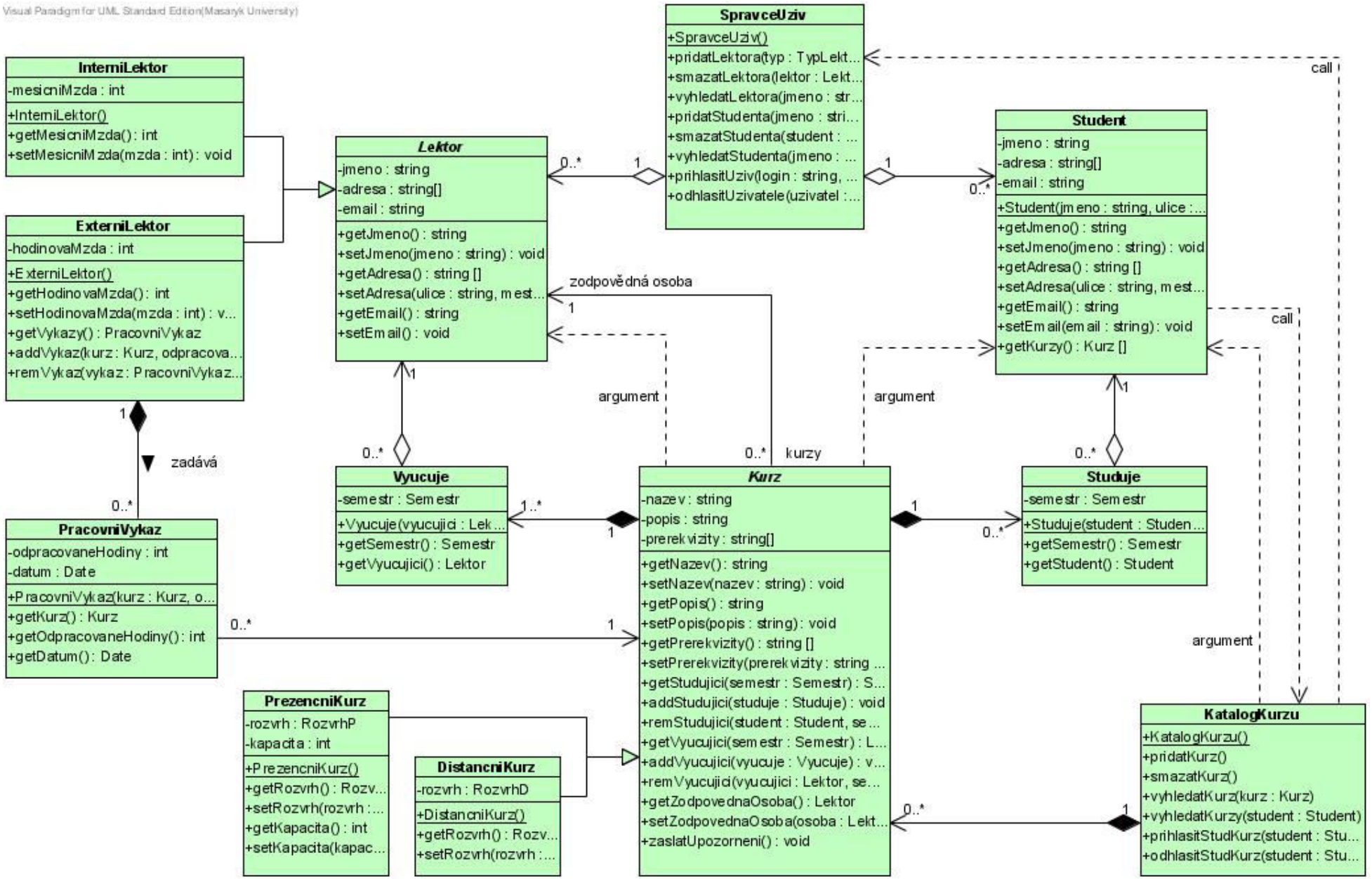
Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/tut/siscz/06_SlovniAnalyza.html

Príklad výstupu kroku 2 (Analytic CD)



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/06/06_Studium_ClassAnalyza.jpg

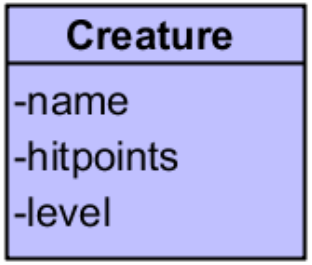
Príklad výstupu kroku 3 (Design CD)



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/10/10_Studium_ClassNavrh.jpg

Trieda

- **Trieda** – Vzor množiny objektov, ktoré zdieľajú rovnaké vlastnosti a chovanie (atribúty a metódy)

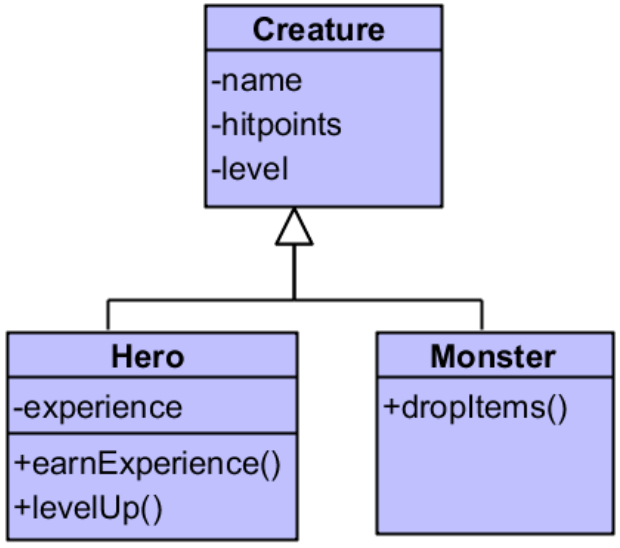



```
public class Creature {
    private int name;
    private int hitpoints;
    private int level;
}
```

- Inštancia – Konkrétny objekt vytvorený na základe triedy

Dedičnosť

- **Dedičnosť** – Hierarchický vzťah predok–potomok

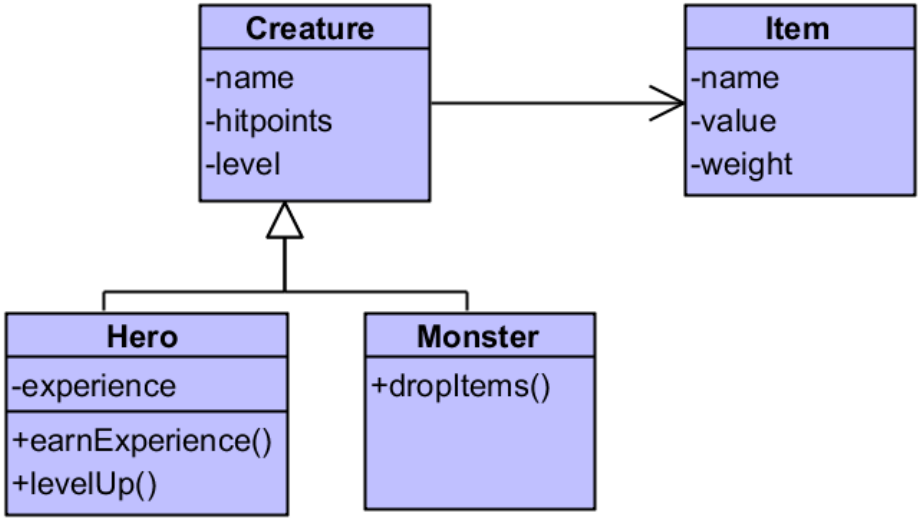



```
public class Hero extends Creature {
    private int experience;

    public void earnExperience() {...}
    public void levelUp() {...}
}
```

Asociácia

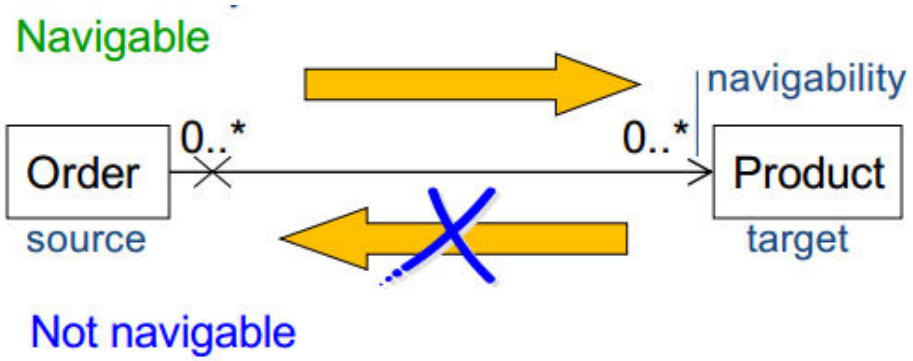
- **Asociácia** – Vzťah medzi inštanciami daných tried




```
public class Creature {
    private int name;
    private int hitpoints;
    private int level;
    private Item item;
}
```

Navigovateľnosť

- Udáva, ktorá trieda bude obsahovať atribút druhej triedy
  - A $\rightarrow$ B: trieda A obsahuje atribút triedy B
  - A $\leftrightarrow$ B $=$ A – B: obojsmerný vzťah

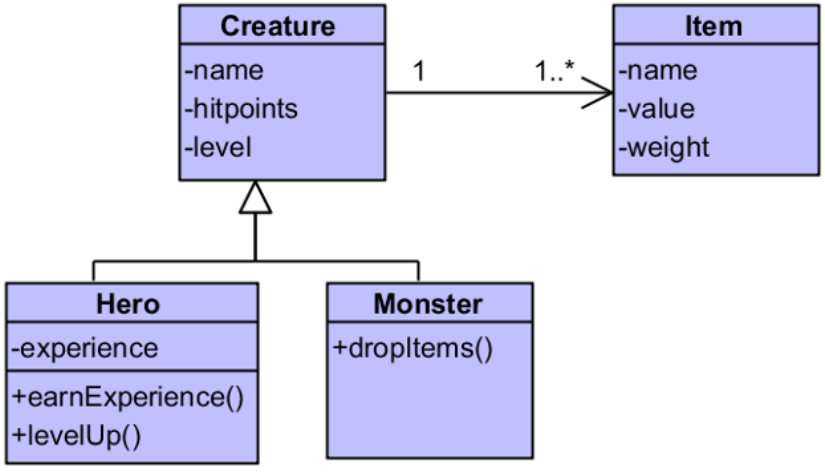


Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

- Objekt „objednávka“ ukladá zoznam produktov
- Objekt „produkt“ neukladá zoznam objednávok

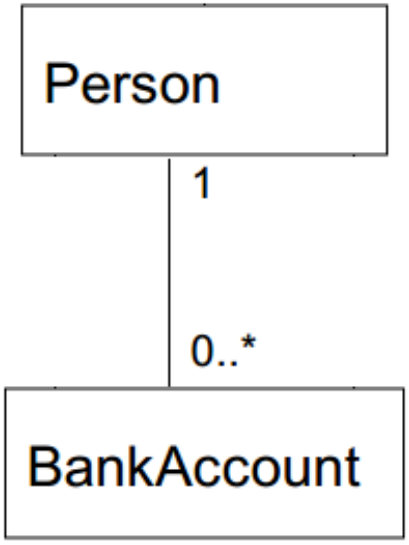
Násobnosť

- Násobnosť (multiplicita) – Počet inštancií tried, ktoré sa môžu zúčastniť vzájomného vzťahu




```
public class Creature {
    private int name;
    private int hitpoints;
    private int level;
    private List<Item> items;
}
```

Násobnosť

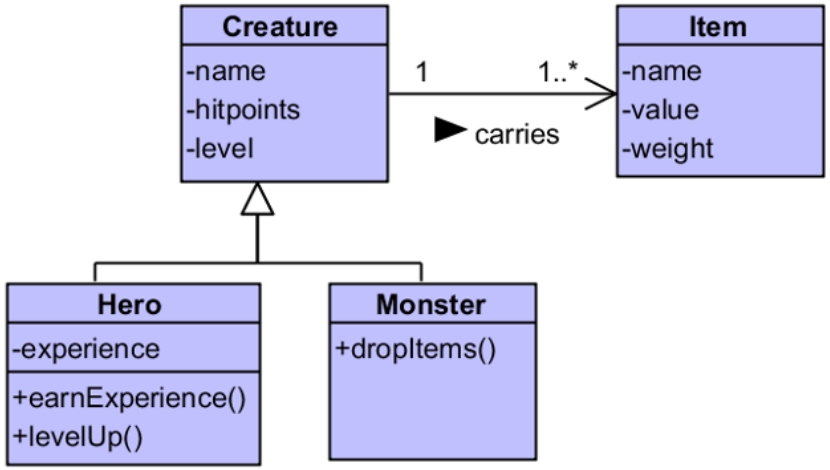


Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

- 1 osoba môže mať 0 až n bankových účtov
- 1 bankový účet prislúcha práve 1 osobe

Názov asociácie

- Pomenovanie vzťahu

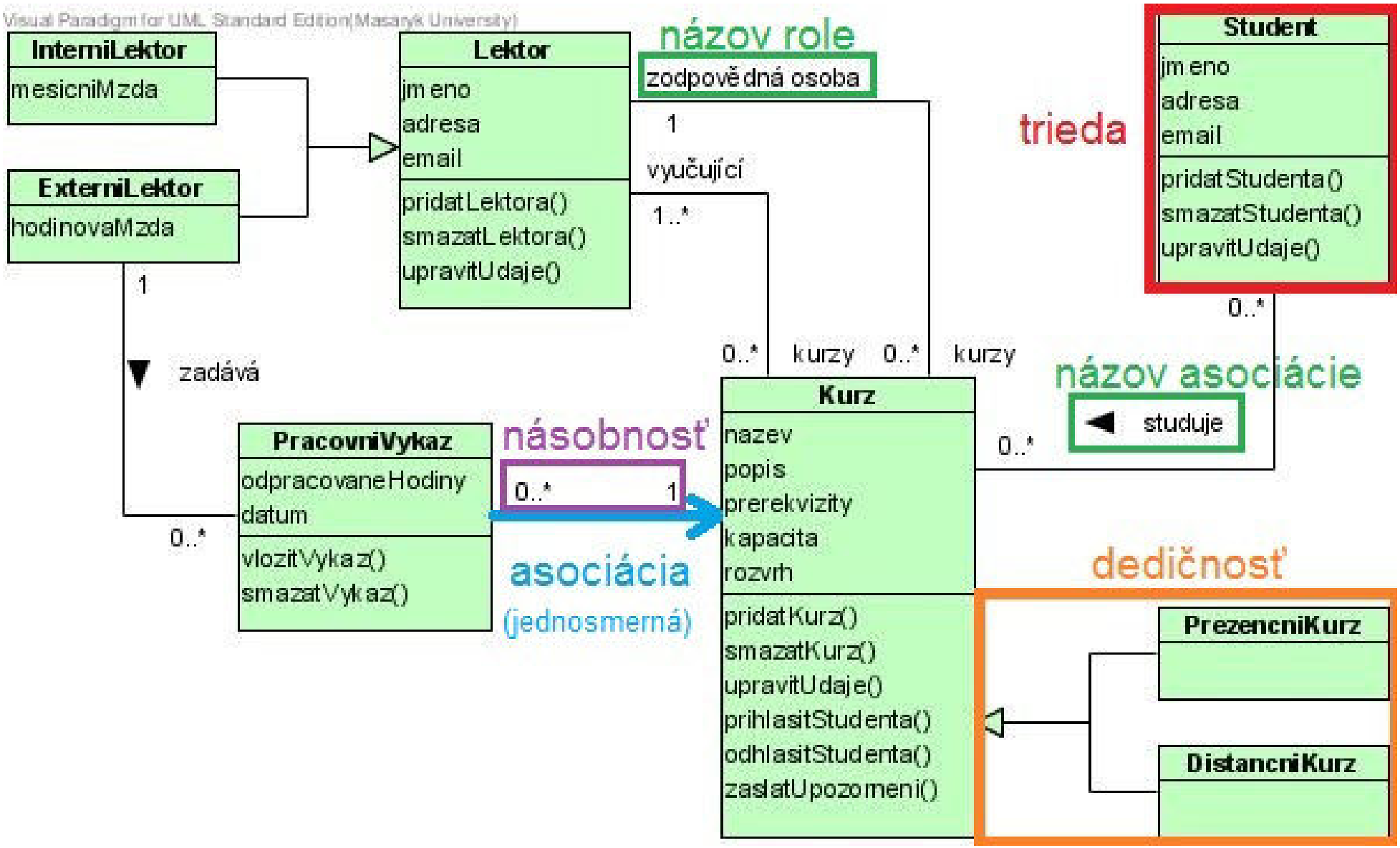



```
public class Creature {
    private int name;
    private int hitpoints;
    private int level;
    private List<Item> items;
}
```

Analytic Class Diagram – prvky – zhrnutie

- **Trieda** – Vzor množiny objektov, ktoré zdieľajú rovnaké vlastnosti a chovanie (atribúty a metódy)
  - Inštancia – Konkrétny objekt vytvorený na základe triedy
- **Dedičnosť** – Hierarchický vzťah predok–potomok
  - Potomok dedí atribúty a operácie od predka
  - Môže mať aj svoje vlastné atribúty a operácie
- **Asociácia** – Vzťah medzi inštanciami daných tried
  - Inštancia triedy A je atribútom inštancie triedy B!
- Navigovateľnosť udáva smer asociácie
- Násobnosť (multiplicita) – Počet inštancií tried, ktoré sa môžu zúčastniť vzájomného vzťahu
  - 1:1, 1:N, (N:1), M:N, 0:1, 0:N, (N:0)
- Názov asociácie / role
  - Pre daný vzťah vybrať jedno, odporúča sa názov asociácie

Analytic Class Diagram – ukážka



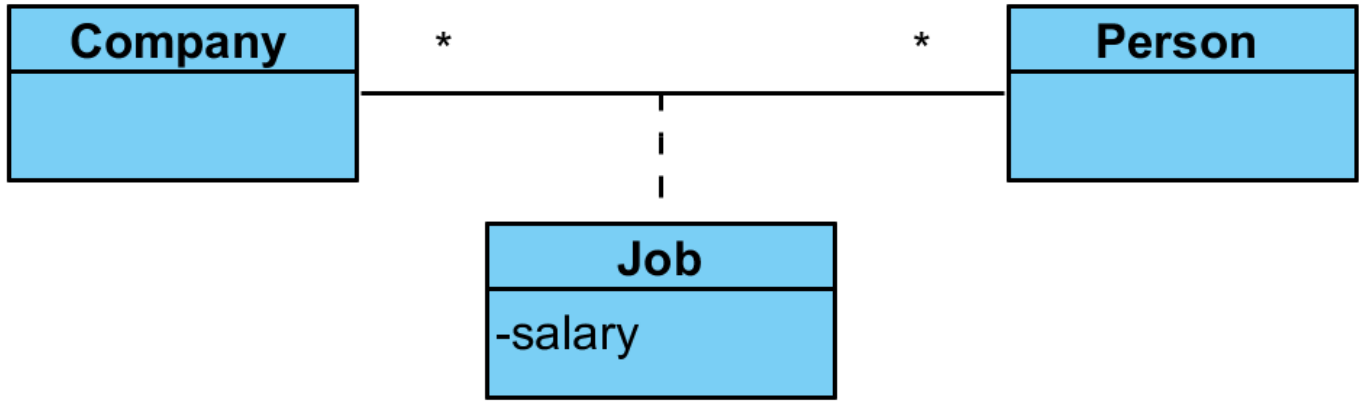
Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/06/06_Studium_ClassAnalyza.jpg

Asociačná trieda

- Rozbíja vzťah násobnosti M:N (obr. 1)
- Zápis: obr. 2 $\equiv$ obr. 3 (rovnaká informácia)

1.

2.



3.

Krok 1: Slovná analýza – postup

Analýza podstatných mien a slovies:

- Tím zhromaždí dostupné zdroje (textová špecifikácia, dokumentácia prípadov použitia, . . .)
- Podstatné mená = kandidáti na triedy alebo atribúty
- Slovesá a slovesné väzby = kandidáti na metódy tried
- https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/tut/siscz/06_SlovniAnalyza.html

CRC (class, responsibilities, collaborators) analýza:

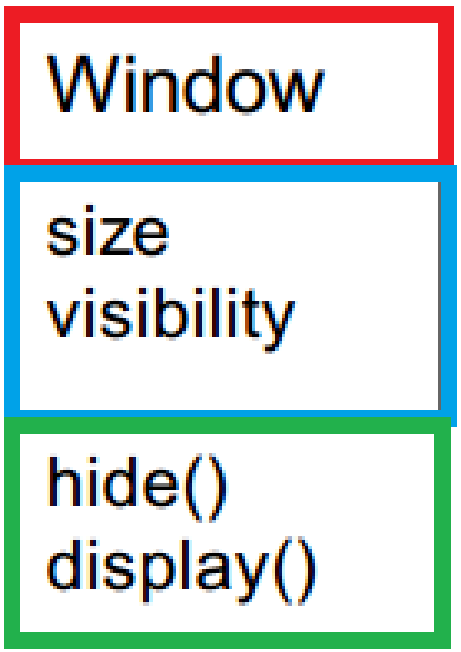
- Tímový brainstorming
- Členovia zapisujú na lístky kandidátne triedy – ich názov, zodpovednosti (metódy) a spolupracovníkov (iné triedy)

Krok 2: Analytic Class Diagram – postup

1. Nájdite triedy, ich základné atribúty a metódy a ich spolupracovníkov
2. Určte dedičnosť medzi triedami (ak treba)
3. Zachyťte vzťahy medzi triedami pomocou asociácií
4. Pomenujte asociácie alebo role
5. Určte násobnosti a navigovateľnosti asociácií
6. Skontrolujte si diagram
7. Doplňte ďalšie atribúty, metódy a závislosti
8. Prehľadne usporiadajte prvky diagramu

Analytic Class Diagram – konvencie

- Názvy tried sú podstatné mená v jednotnom čísle
  - Ideálne v angličtine zapísané UpperCamelCase notáciou
- Názvy atribútov sú podstatné mená v jednotnom čísle
  - S malým písmenom na začiatku
- Názvy metód sú slovesá v neurčitku / slovesné väzby
  - S malým písmenom na začiatku



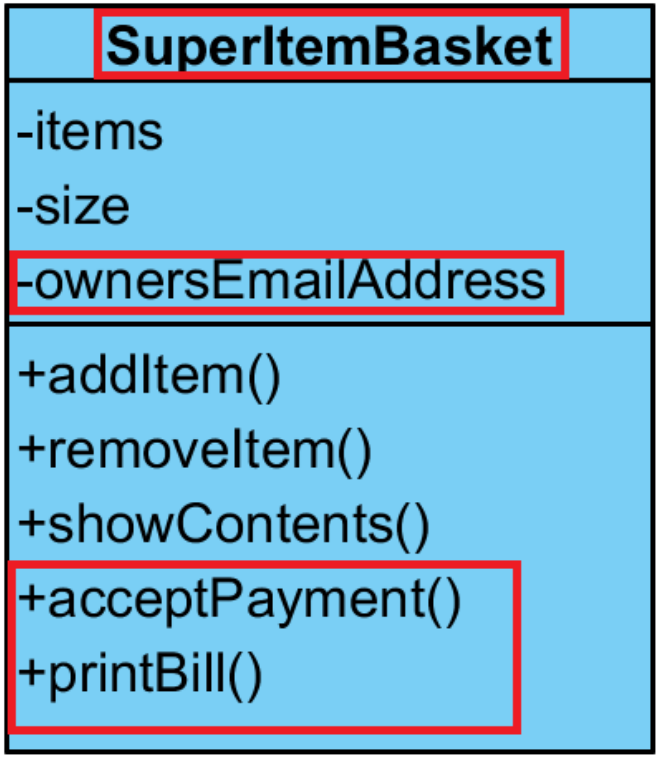
Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

Analytic Class Diagram – tipy

Dobrá analytická trieda:

- Názov vyjadruje jej účel
- Má len dôležité atribúty, ktoré chceme modelovať
- Má jednu zodpovednosť (high cohesion)
  - 3–5 metód

Zlá analytická trieda:

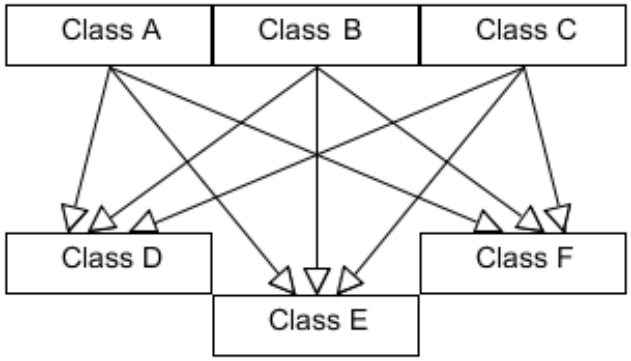


Analytic Class Diagram – tipy

Dobrý analytický diagram:

- Triedy spolu komunikujú
- Ale majú medzi sebou málo väzieb (low coupling)

Zlý analytický diagram:



Zdroj:
https://ayumilovedesignpattern.wordpress.com/

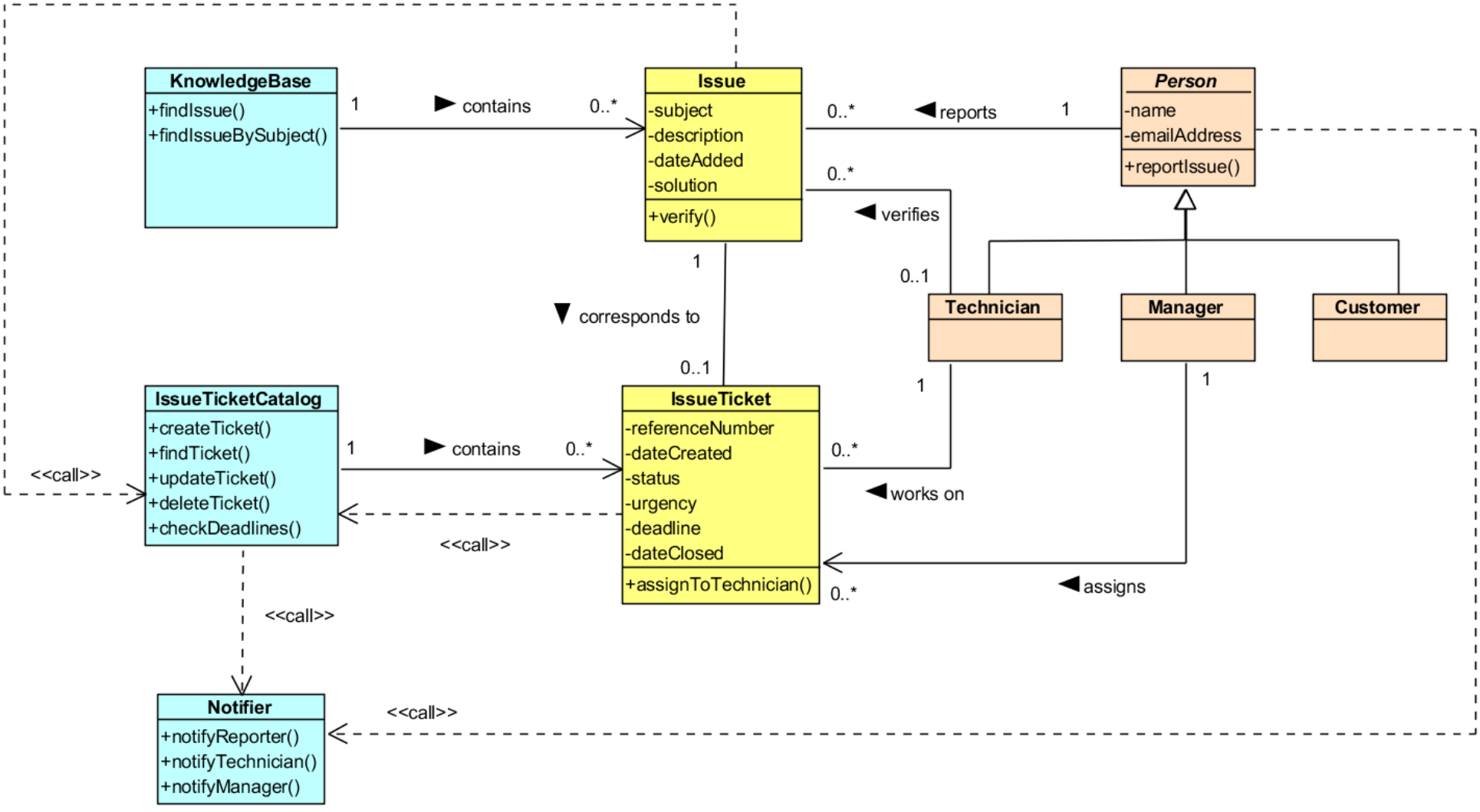
Tiež:

- 10–20 tried
- Dedičnosť len vtedy, ak je to nutné

Tiež:

- Veľa malých tried
- Málo veľkých tried
- Zložitá dedičnosť

Príklad riešenia



PB007 Softwarové inženýrství I

Cvičenie 7 – State Machine Diagram

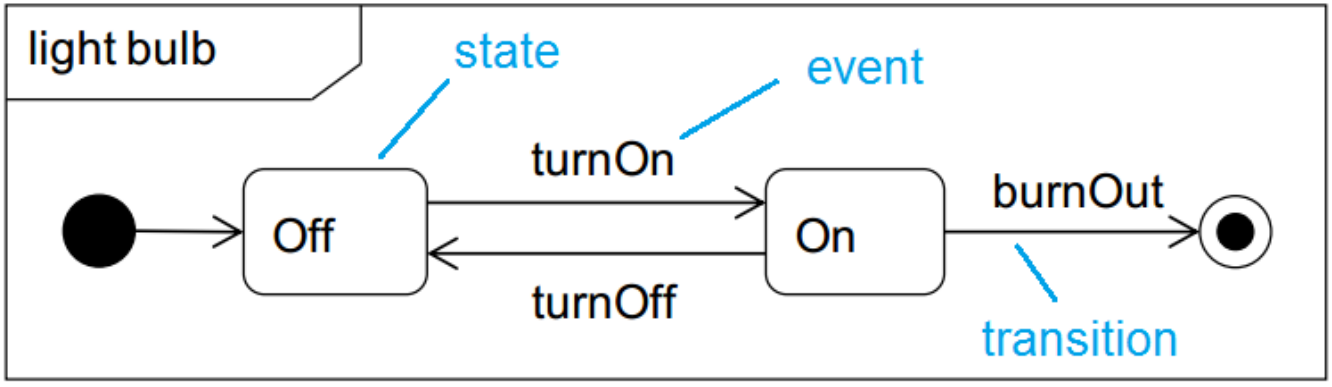
Valdemar Švábenský

Fakulta informatiky, Masarykova univerzita, Brno

27. októbra 2015

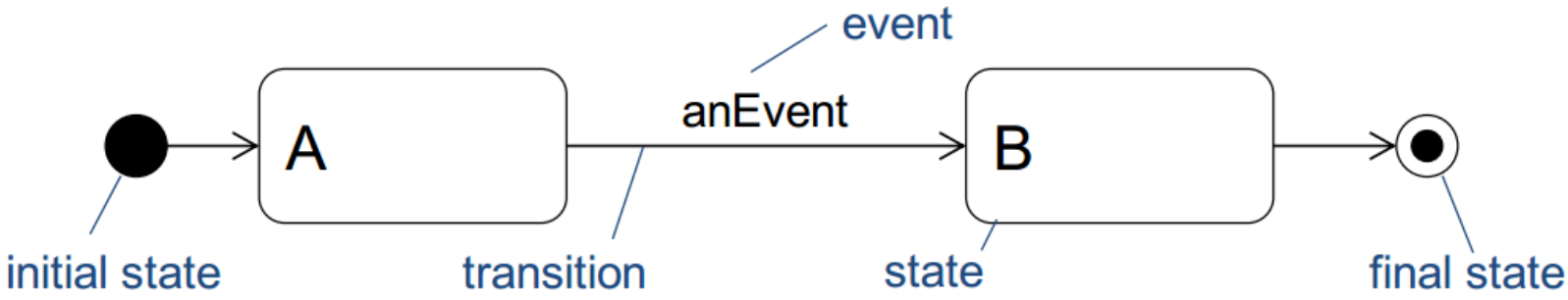
Stavový diagram (State Machine Diagram)

- Popisuje životný cyklus (dynamické chovanie v priebehu času) vybraného objektu
- Objekt = trieda, prípad použitia, celý systém, ...
- Životný cyklus je modelovaný ako postupnosť:
  - stavov (*states*),
  - prechodov (*transitions*) medzi stavmi a
  - udalostí (*events*), ktoré zmeny stavu vyvolávajú



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

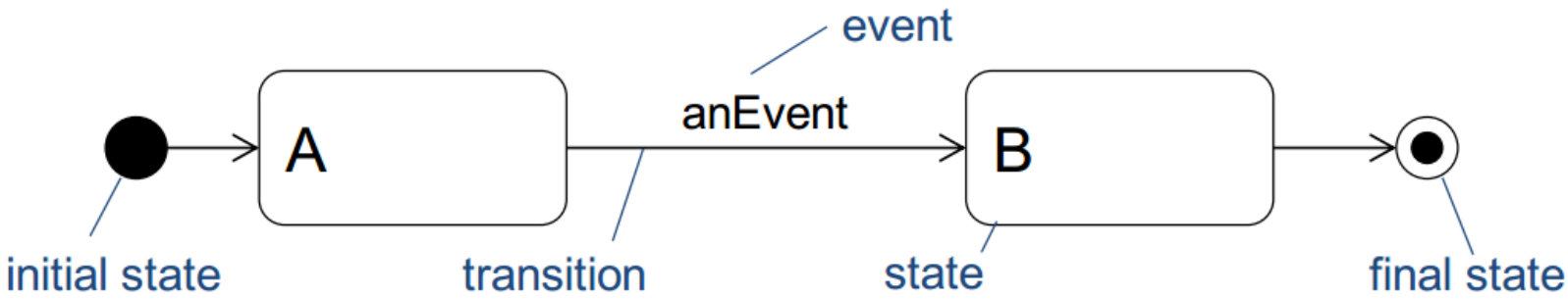
Stavy (states)



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

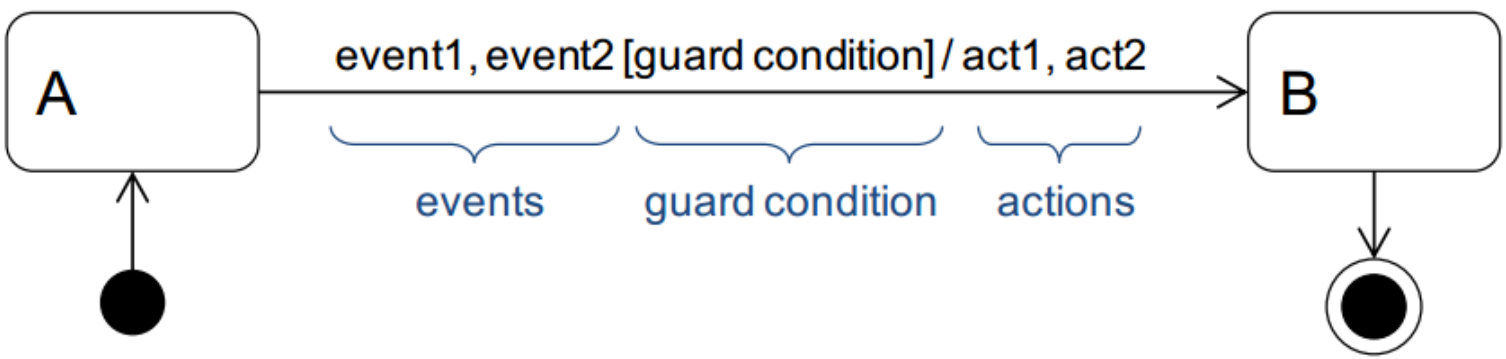
- Stav reprezentuje významnú situáciu v živote objektu
- Je určený hodnotami atribútov, vzťahmi s inými objektami a vykonávanými aktivitami
- Špeciálne typy stavov:
  - Každý stavový diagram má iniciálny stav
  - Ak stavy necyklia do nekonečna, každý stavový diagram by mal mať koncový stav
  - Zlučovacie a rozhodovacie pseudostavy, pozri ďalej

Prechody (transitions)



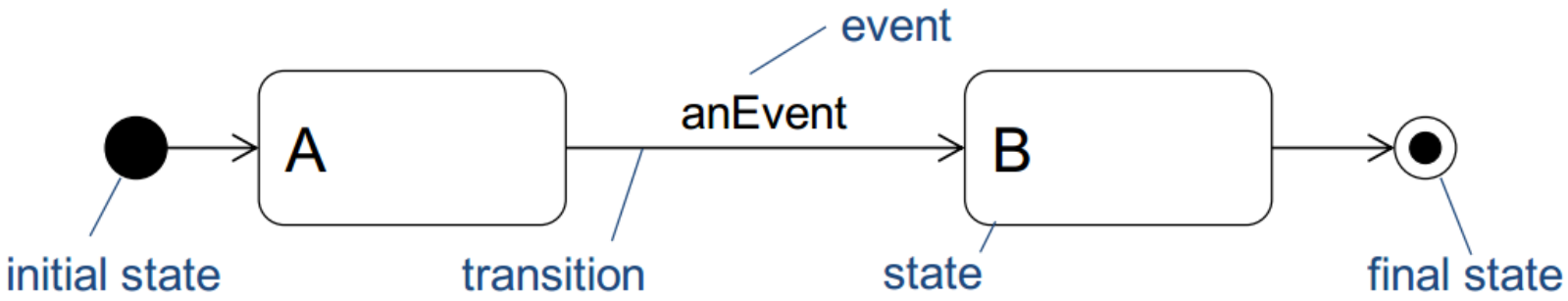
Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

- Prechod určuje, ako sa dostať z jedného stavu do druhého
- Syntax: `udalosť[podmienka]/akcia`
- Pri výskyte `udalosti`, ak je splnená `podmienka`, vykonaj `akciu` a prejdi do nového stavu



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

Udalosti (events)

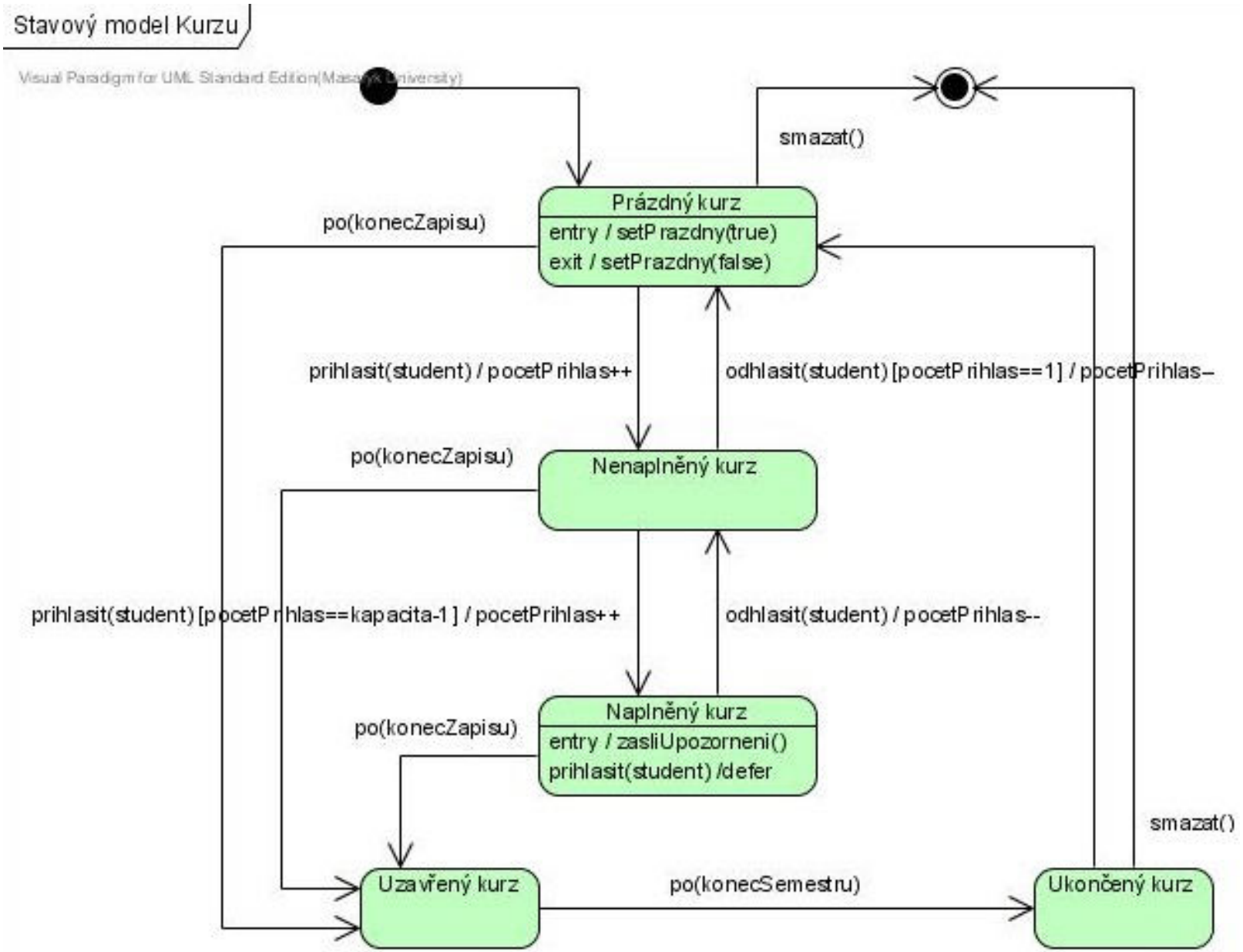


Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

- Udalosť je podnet, na ktorý môže objekt reagovať zmenou stavu alebo vykonaním nejakej akcie
- Externá (na prechodoch) alebo interná (vnútri stavov)

1. Call event – volanie operácie objektu
2. Signal event – asynchrónne poslanie a príjem signálu od jedného objektu k druhému
3. Change event – logická podmienka, ktorá spôsobí prechod, keď sa jej hodnota zmení z `false` na `true`
4. Time event – časový výraz: udalosť nastane v určitú dobu (`when()`) alebo po určitej dobe (`after()`)

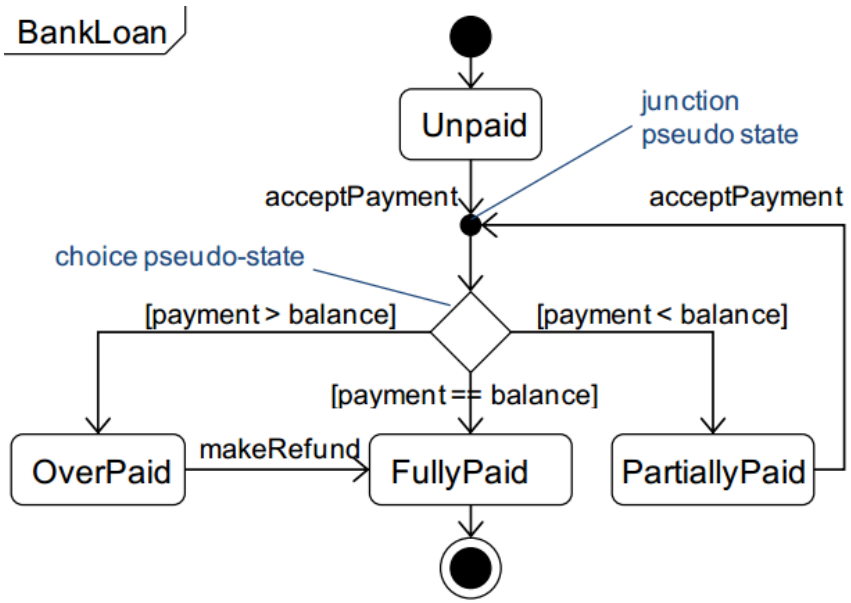
Stavový diagram – ukážka 1



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/tut/siscz/05_Studium_StateMachnine.jpg

Rozhodovacie a zlučovacie pseudostavy

- Rozhodovací pseudostav (*choice pseudo-state*) – spôsobí prechod z jedinej vstupnej hrany do jednej z výstupných
- Zlučovací pseudostav (*junction pseudo-state*) – spája viacero vstupných hrán do jednej výstupnej



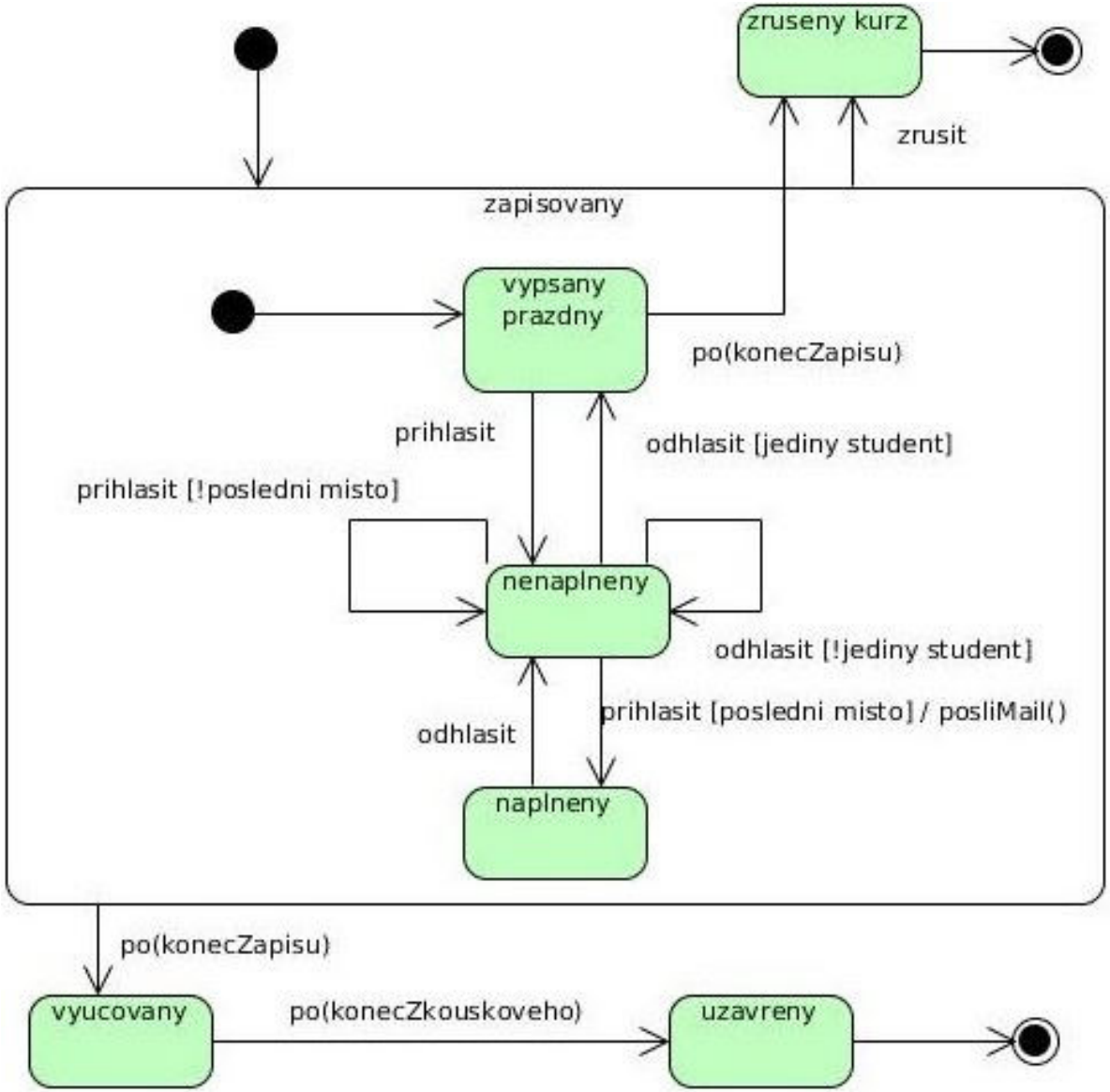
Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/04-ObjectOrientedAnalysis.pdf

- Podobné Decision / Merge Node v diagrame aktivít
- Stavový diagram je podobný diagramu aktivít, líši sa však sémantikou a účelom – je určený k modelovaniu objektov

Zložené stavy

- Zložený stav je stav, ktorý obsahuje vnorené stavy.
- **Jednoduché** zložené stavy
  - Tvorí ich 1 región
  - Sú vhodné na zachytenie dedičnosti medzi stavmi
- **Ortogonálne** zložené stavy
  - Tvorí ich 2 a viac regiónov
  - Každý z nich obsahuje vnorený stavový diagram, ktorých vykonávanie prebieha paralelne

Stavový diagram – ukážka 2



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/05/05_Studium_StateMachine-inher.jpg

Tipy

- Všetky stavy okrem počiatočného a koncového by mali mať vstupný a aj výstpný prechod
- Diagram by mal byť čitateľný zľava doprava a zhora nadol
  - Počiatočný stav v ľavom hornom rohu, koncový stav v pravom dolnom rohu
- Udalosť pomenujte ako sloveso v trpnom rode
- Skúste využiť dedičnosť medzi stavmi a tým zjednodušiť výsledný model

PB007 Softwarové inženýrství I

Cvičenie 8 – Entity-Relationship Diagram

Valdemar Švábenský

Fakulta informatiky, Masarykova univerzita, Brno

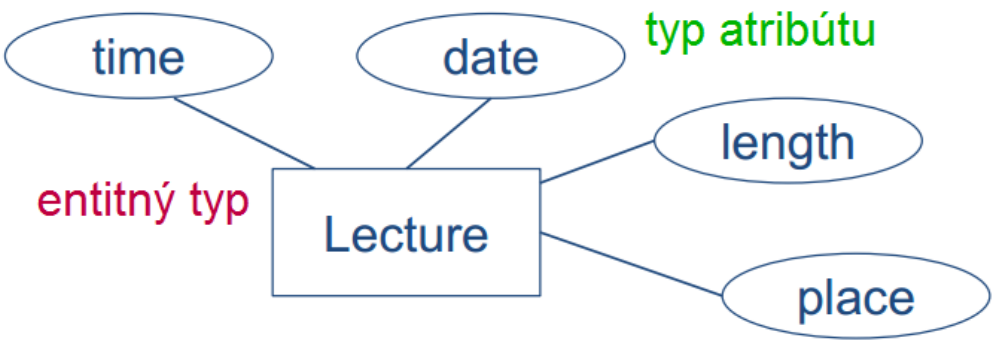
3. novembra 2015

Entitne-relačný diagram (ERD)

- Dátový model je sada nástrojov pre popis dát (ich syntaxe a sémantiky), vzťahov medzi dátami a podmienok, ktoré platia na dátach
  - Slúži pre návrh dátovej štruktúry
- Entitne-relačný model je dátový model, ktorý reprezentuje logickú štruktúru databázy
  - Odvádza sa z neho **relačná schéma databázy**
- ERD je grafickým vyjadrením entitne-relačného modelu
- Základné zložky ERD:
  - **Entity** (resp. entitné typy)
  - **Atribúty** (resp. typy atribútov)
  - **Vzťahy** (resp. vzťahové typy)

ERD – entita

- Entita je **objekt**, ktorý existuje, je odlíšiteľný od ostatných objektov a je v navrhovanom systéme potrebný
  - Abstraktná (výuka) aj konkrétna (študent)
  - Popísaná svojim **názvom** a množinou **atribútov**
  - Jednoznačne identifikovaná **kľúčom**
- Entitný typ je vzor skupiny entít, ktoré zdieľajú rovnaké typy atribútov
  - Entita je „inštanciou“ entitného typu
  - Entitná množina je skupina všetkých entít daného entitného typu v databáze v aktuálnom čase

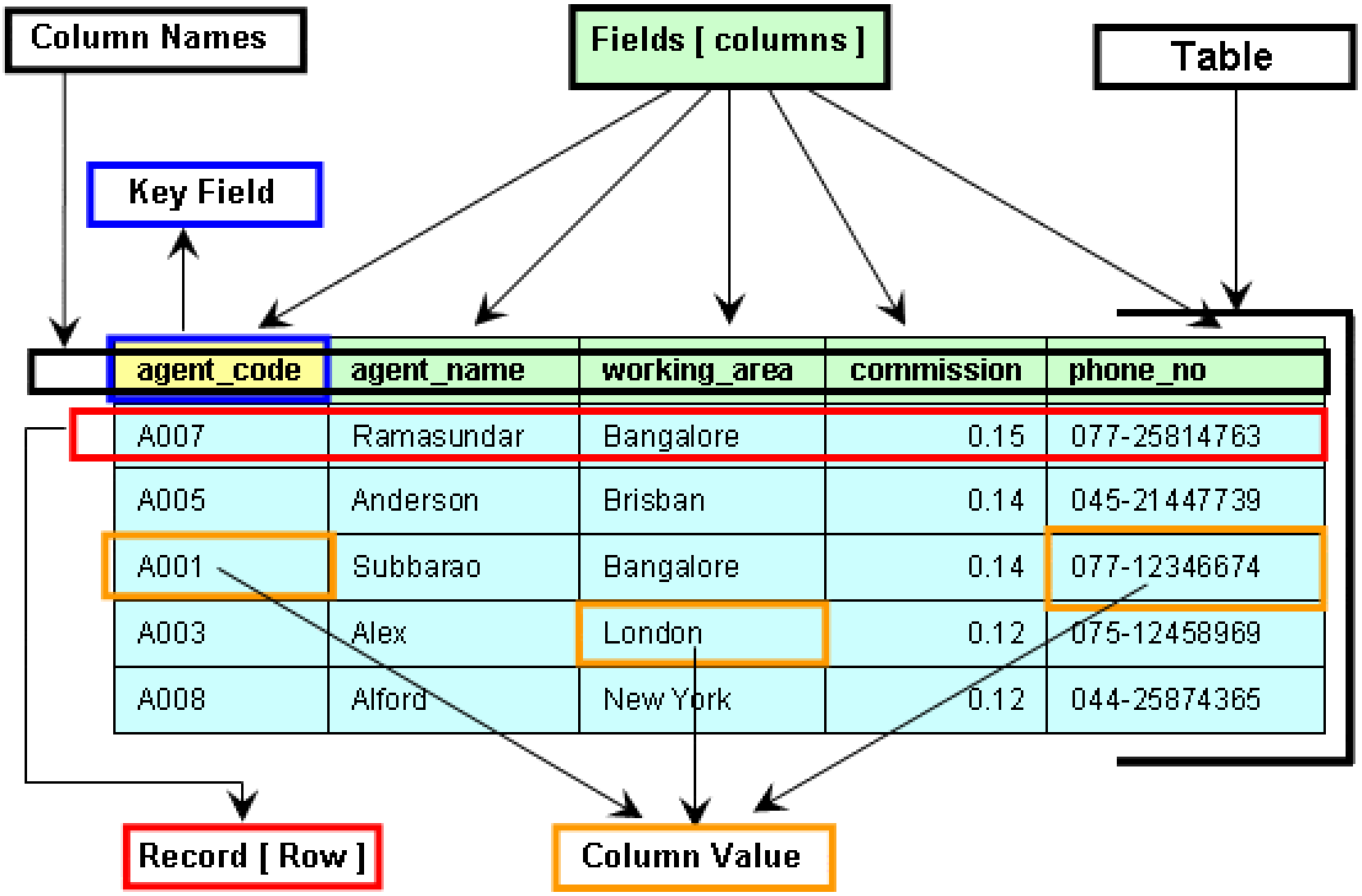


Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/05-StructuredAnalysis.pdf

ERD – atribút

- Atribút je popisná **vlastnosť**; informácia o entite alebo vzťahu, ktorej hodnotu uchovávame a používame v systéme
- Každý atribút má dátový typ
- Druhy atribútov:
  - Jednoduché (napr. meno) a zložené (napr. dátum)
  - S jednoduchou hodnotou (*single-valued*, napr. meno) a s násobnou hodnotou (*multivalued*, napr. telefónne číslo)
  - Nulové (*null*) (napr. osoba nemá telefón)
  - Odvodené (*derived*) (napr. vek z dátumu narodenia)

Databázová tabuľka



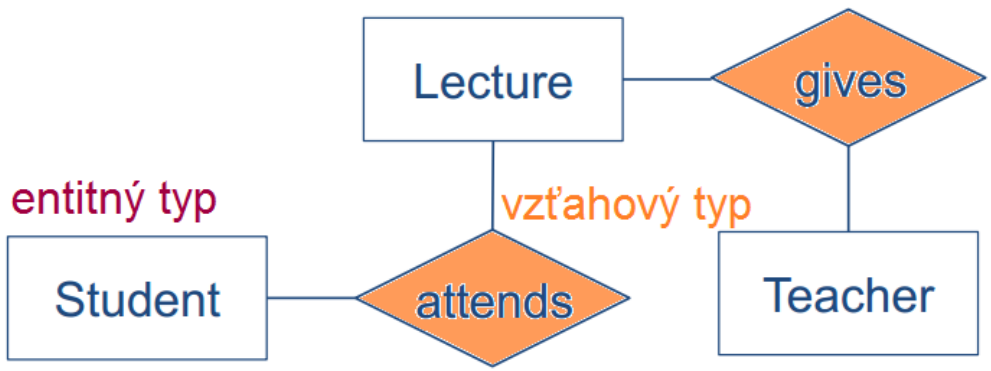
Zdroj: http://www.w3resource.com/sql/sql-basic/the-components-of-a-table.php

ERD – kľúč

- **Kľúč** je časť relačnej schémy, podmnožina atribútov, ktorá identifikuje entitu (záznam, n-ticu v tabuľke)
- **Superkľúč** je identifikátor entity dostatočný pre jednoznačnú identifikáciu
- **Kandidátny kľúč** je minimálny superkľúč, tzn. po odobraní akejkoľvek množiny atribútov by už neidentifikoval entity jednoznačne
- **Primárny kľúč** je jeden zvolený kandidátny kľúč
  - Obvykle umelo vytvorené celočíselné ID, ktoré sa automatizovane inkrementuje
- **Cudzí kľúč** je množina atribútov v entite B, ktorá je superkľúčom v inej entite A
  - Obvykle kandidátny alebo primárny kľúč entity A

ERD – vzťah

- Vzťah je spojenie medzi niekoľkými entitami, o ktorom uchovávame informácie
- Vzťahový typ[1] je vzor vzťahov rovnakého druhu (môže mať atribúty)



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/05-StructuredAnalysis.pdf

[1] Zdroj: http://web.cse.ohio-state.edu/~gurari/course/cse670/cse670Ch2.xht

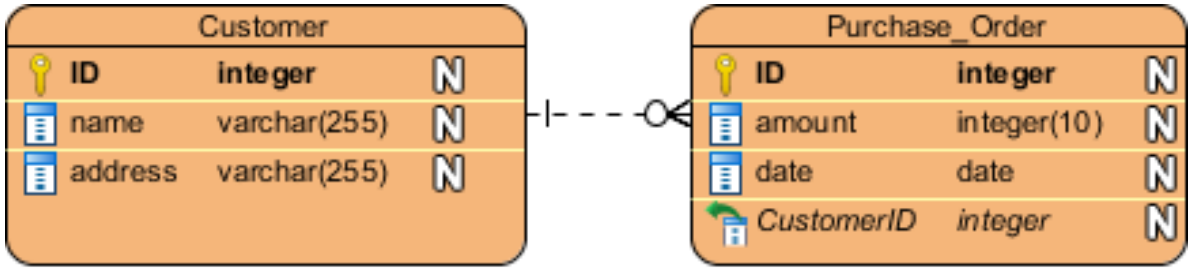
ERD – popisy vzťahu

- **Rola** – úloha, ktorú nadobúda entita vo vzťahu
  - Vo vzťahu *manželstvo* medzi entitami typu *osoba* a *osoba* získavajú entity role muž a žena[2]
- **Stupeň vzťahu** – počet množín entít, ktoré sú súčasťou množiny vzťahu (obvykle unárny/binárny, príp. ternárny)
- **Kardinalita** – počet prvkov množiny; pri modelovaní sa niekedy zamieňa s korektnejším termínom multiplicita
- **Násobnosť vzťahu** (multiplicita) – počet entít, ktoré sa môžu zúčastniť vzťahu
  - Udáva dolnú a hornú kardinalitu
  - Typy: 0:1, 1:1, 0:N, 1:N, M:N
  - Pri násobnosti M:N sa počas implementácie vytvára stredná (tzv. asociačná) entita, ku ktorej majú predošlé dve entity vzťah 1:N

[2] Prípadne muž–muž alebo žena–žena

ERD – cudzí kľúč detailne

- Vo vzťahu 1:N je cudzí kľúč na strane „N“; resp. ukazuje na stranu „1“:

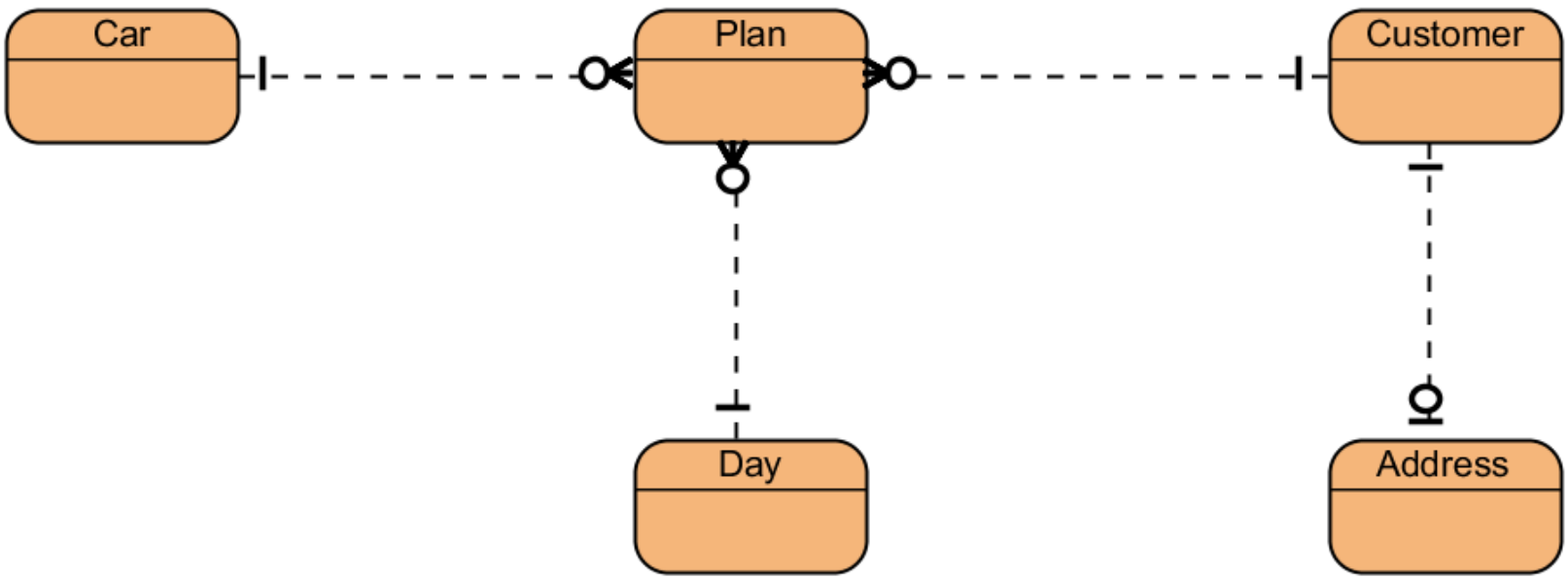


Zdroj: http://www.visual-paradigm.com/tutorials/compare-logical-physical-erd.jsp

- Vo vzťahu 1:1 je umiestnenie cudzieho kľúča na návrhárovi; prípadne sa dané dve tabuľky môžu zlúčiť do jednej

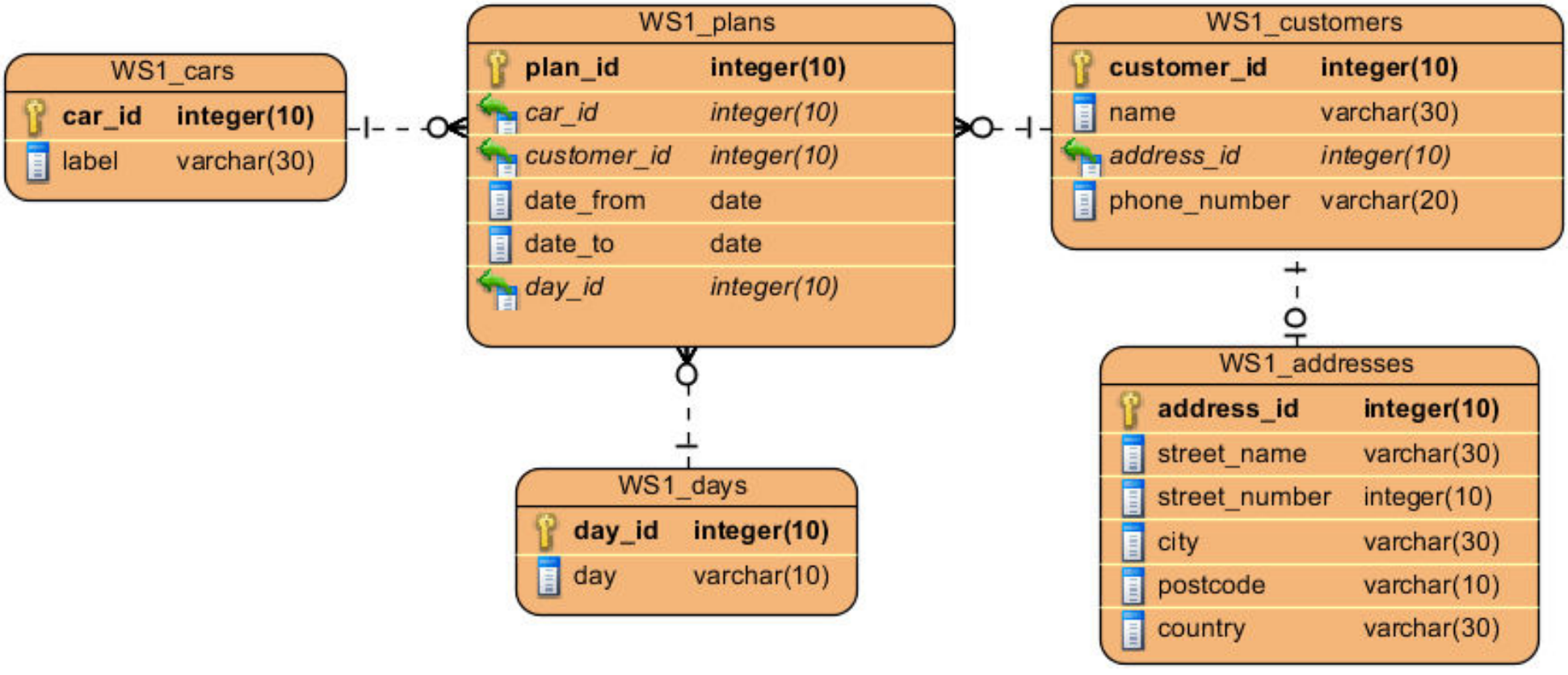
ERD – konceptuálny model – ukážka

- Spoločnosť pre rozvoz tovaru
- Ku každému `autu` je priradený `plán`, ktorý určuje, k akému `zákazníkovi` na akej `adrese` bude auto jazdiť daný `deň`
- Zákazníka `Z` obsluhuje Auto `A` v `i`-tý deň v týždni



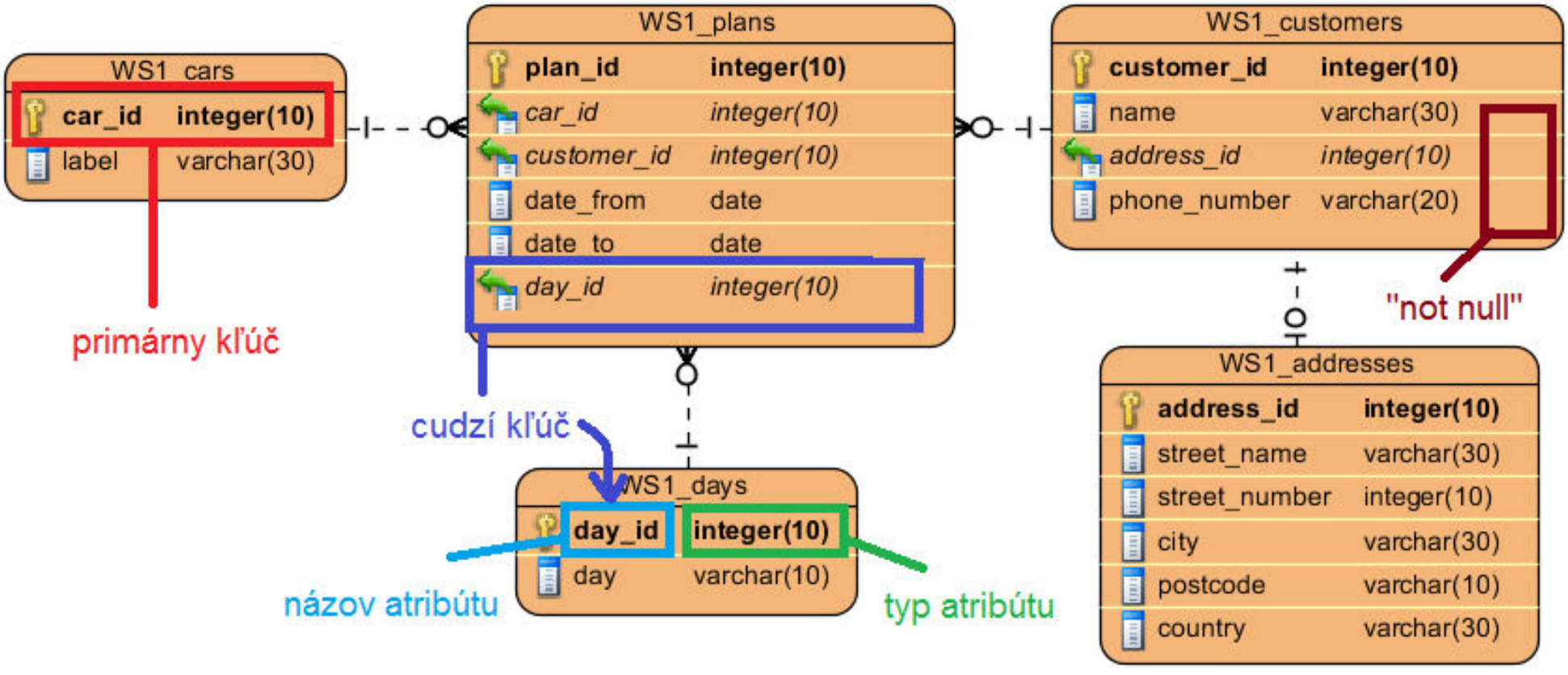
- Model definuje: entity, vzťahy, násobnosti

ERD – implementačný, fyzický model – ukážka



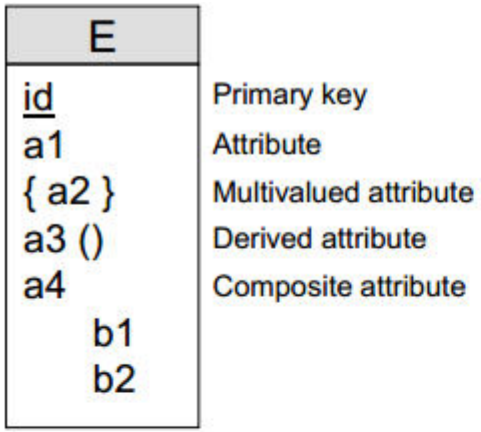
- Model definuje: entity, atribúty a ich dátové typy, vzťahy, násobnosti, kľúče, integritné obmedzenia

ERD – implementačný, fyzický model – ukážka



- Model definuje: entity, atribúty a ich dátové typy, vzťahy, násobnosti, kľúče, integritné obmedzenia

ERD – grafické vyjadrenie

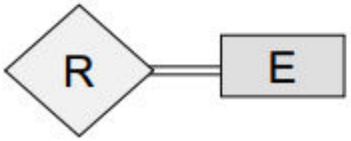


Entity set

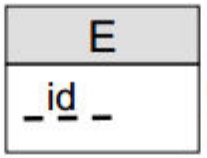
Many-to-many relationship set

Many-to-one relationship set

One-to-one relationship set

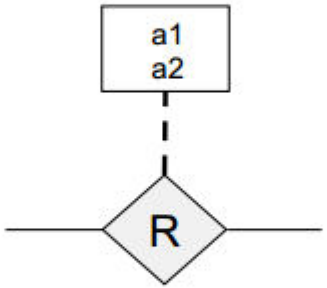


Total participation of entity set in relationship

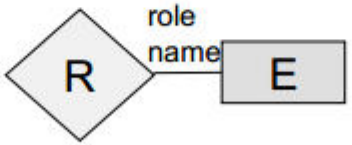


Weak entity set

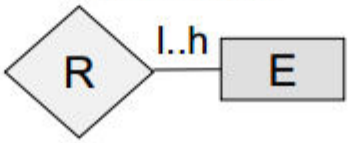
Identifying relationship set for weak entity set



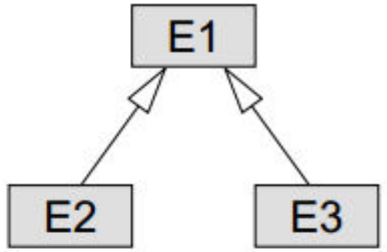
Relationship attributes



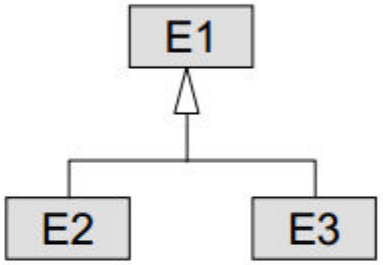
Role indicator



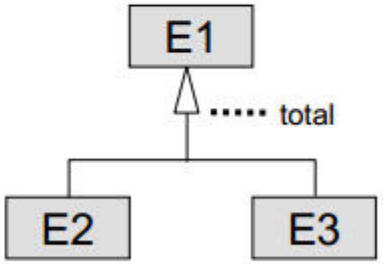
Cardinality limits



Generalization / specialization (overlapping)



Disjoint generalization



Total disjoint generalization

Source: unknown

Objektovo-relačné mapovanie (ORM)

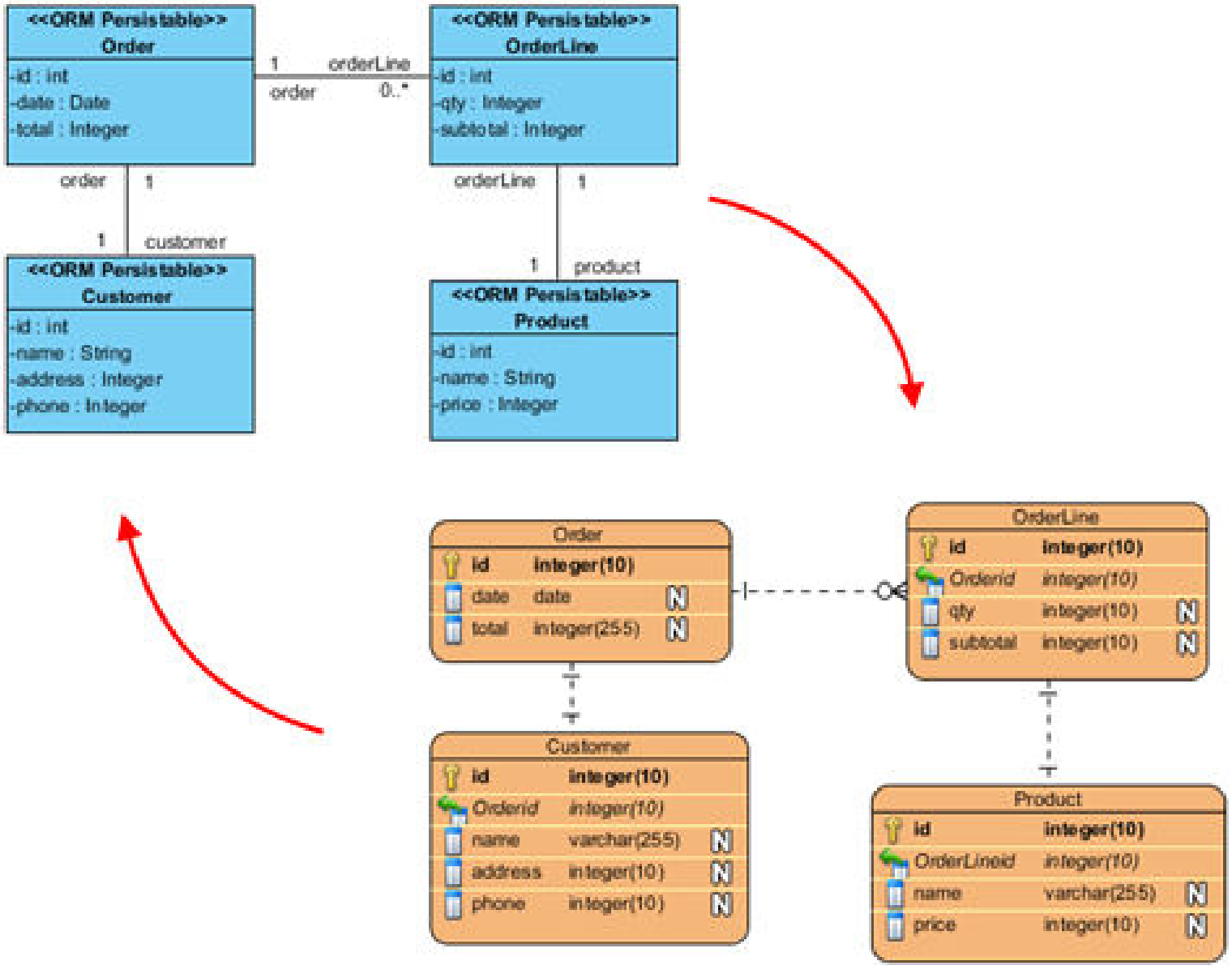
ORM je technika konverzie dát medzi relačnou databázou a objektovo orientovaným jazykom

- Perzistentná trieda $\equiv$ entitný typ (tabuľka v DB)
- Objekt $\equiv$ entita (riadok v tabuľke)
- Atribúty triedy $\equiv$ atribúty entity (stĺpce tabuľky)
- Asociácia tried $\equiv$ relácia (prepojenie tabuliek cudzími kľúčmi)
- Dedičnosť je možné riešiť tromi spôsobmi: mapovanie 1:1, zahrnutie do nadtriedy, rozpustenie do podtried

Poznámky:

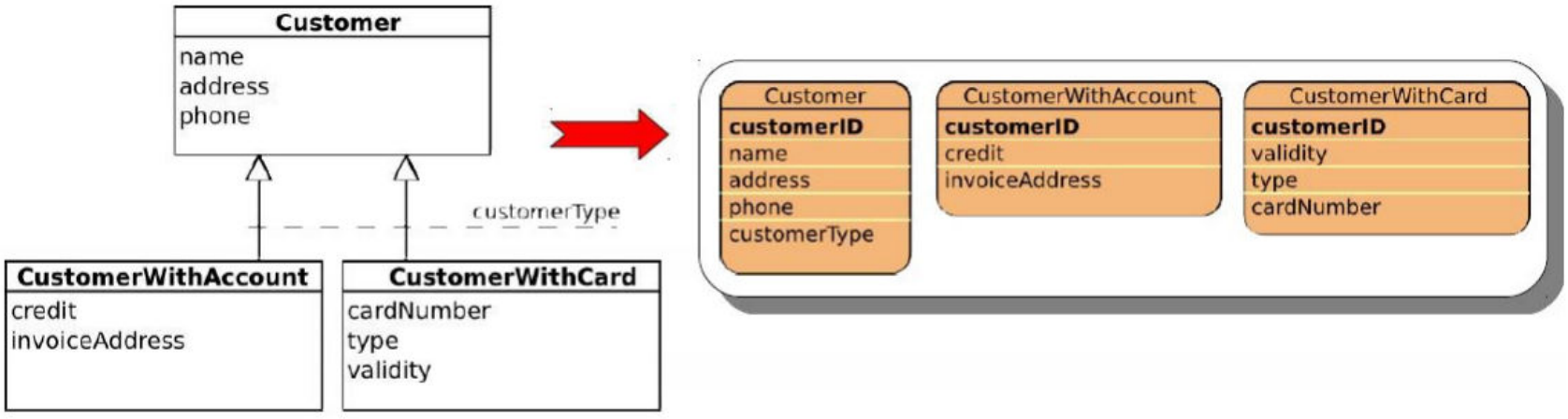
- Jedna trieda môže byť mapovaná na viac tabuliek
- Viac tried môže byť mapovaných do jednej tabuľky
- Nie všetky triedy musia byť perzistentné

ORM – ukážka



Zdroj: http://www.visual-paradigm.com/VPGallery/datamodeling/SynchronizationToClassDiagram.html

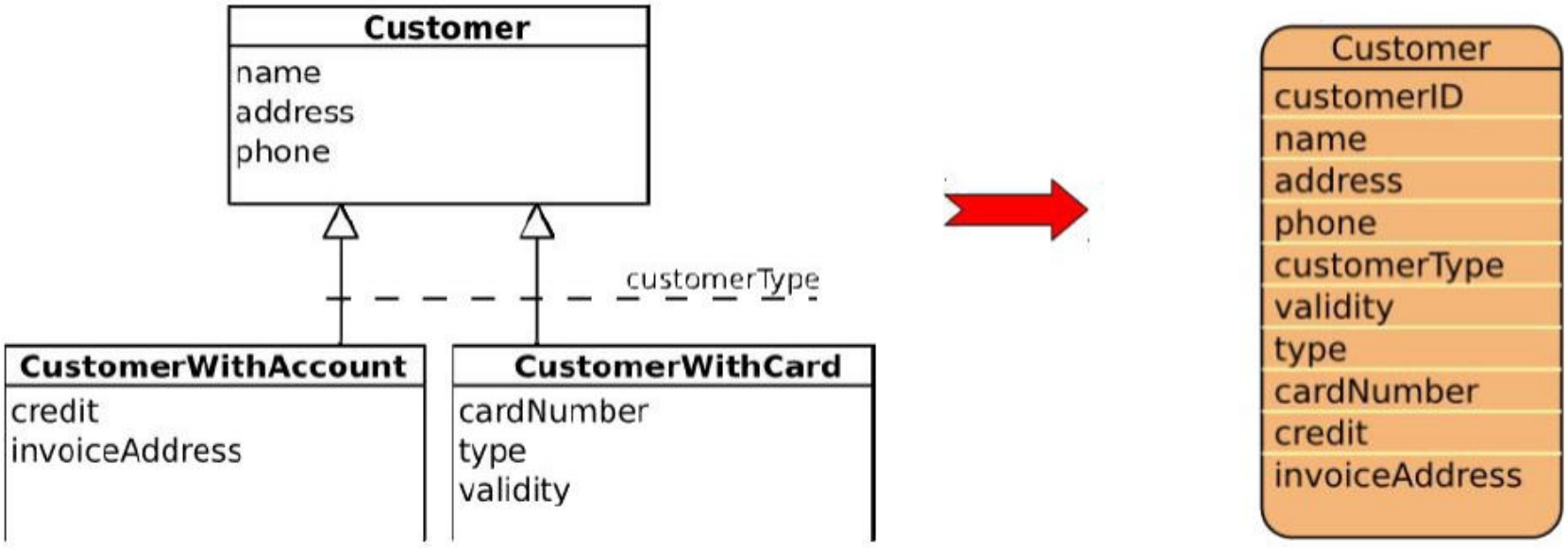
ORM – dedičnosť – a) mapovanie 1:1



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/08/pb007-cvicenie-08.pdf

- Každá trieda sa stáva tabuľkou
- ID ex-nadtriedy = cudzí kľúč v ex-podtriedach
- Jedna inštancia triedy je uložená vo viacerých tabuľkách

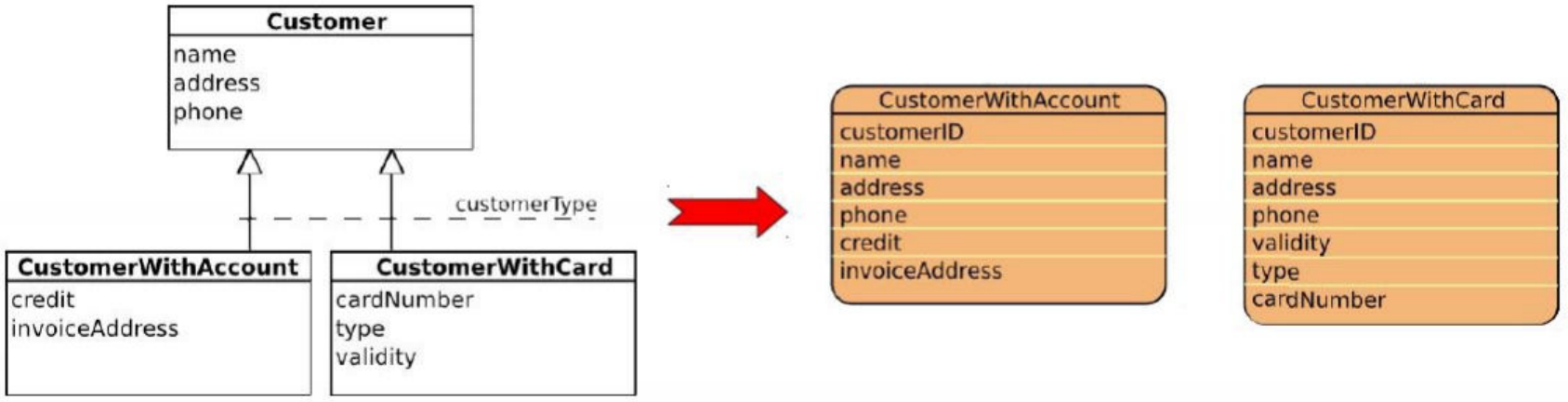
ORM – dedičnosť – b) zahrnutie do nadtriedy



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/08/pb007-cvicenie-08.pdf

- Všetky atribúty podtried sú zahrnuté do jednej tabuľky
- Niektoré atribúty môžu byť `null` – porušenie BCNF
- Vhodné pri menšom počte podtried s málo atribútmi

ORM – dedičnosť – c) rozpustenie do podtried



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/08/pb007-cvicenie-08.pdf

- Atribúty nadtriedy sú prenesené do tabuliek pre všetky neabstraktné podtriedy
- Vhodné, ak:
  - nadtrieda má málo atribútov
  - existuje mnoho podtried
  - podtriedy majú veľa atribútov

Class Diagram vs. ERD

Class diagram	**ERD**
Orientuje sa na dáta aj procesy	Orientuje sa len na dáta
Modeluje štruktúru aj chovanie systému (atribúty, operácie)	Modeluje iba štruktúru dát
Obsahuje rôzne druhy vzťahov (asociácie, agregácie, kompozície, závislosti, generalizácie)	Obsahuje len jednoduché vzťahy (napr. len zriedkavo generalizuje)
Pripomína objekty z reálneho sveta	Pripomína DB tabuľky

1. normálna forma (1. NF)

- Relačná schéma je v 1. NF, keď každý jej atribút je **atomický** (ďalej nedeliteľný; teda jednoduchý a nie zložený ani viachodnotový)
- Relácia *Osoba* sa rozloží na relácie *Osoba* a *Telefon*

Osoba

ID	Jméno	Příjmení	Adresa	Telefony
1	Jan	Novák	Havlíčkova 2 Praha 3	125789654;601258987
2	Petr	Kovář	Svatoplukova 15 Brno	369852147;357951456
3	Pavel	Pavel	Papalášova 25 Kocourkov	546789123;123456789

Osoba

ID	Jméno	Příjmení	Adresa
1	Jan	Novák	Havlíčkova 2 Praha 3
2	Petr	Kovář	Svatoplukova 15 Brno
3	Pavel	Pavel	Papalášova 25 Kocourkov

Telefon

ID_osoby	Cislo
1	125789654
1	601258987
2	369852147
2	357951456
3	546789123
3	123456789

Funkčná závislosť

- Nech R je relačná schéma DB
- Nech $X \subseteq R$, $Y \subseteq R$ sú množiny atribútov
- Potom Y je **funkčne závislé** na X (píšeme $X \rightarrow Y$), keď pre každú povolenú reláciu $r(R)$ platí:
  - ak majú dva ľubovolné prvky tejto relácie rovnaké hodnoty v atribútoch X,
  - potom majú rovnaké hodnoty v atribútoch Y

	SSN	Name	City	State	ZIP
t_1	72163	John Smith	Chicago	IL	90101
t_2	87991	Mark Green	LA	CA	90065
t_3	87891	Mark Smith	LA	CA	90065
t_4	23212	Mary Clarke	LA	CA	90101

Functional Dependencies:

SSN $\rightarrow$ Name, City, State, ZIP

Name $\rightarrow$ SSN, City, State, ZIP

ZIP $\rightarrow$ State, City (violated!)

Zdroj: http://prefex.cs.uwaterloo.ca/projects/probclean/

2. normálna forma (2. NF)

- Relačná schéma je v 2. NF, keď je v 1. NF a každý atribút, ktorý nie je primárny (= nie je súčasťou kandidátneho kľúča), je závislý na celom kľúči (môže byť aj tranzitívne, ale vždy na celom kľúči)
- *Telefon výrobce* nezávisí na celom kľúči, ale len na atribúte *Výrobce*
- Relácia *Sklad* sa rozloží na relácie *Výrobek* a *Výrobce*

Sklad

Název	Výrobce	Telefon Výrobce	Cena	Množství
Mléčná čokoláda	Milka	+420123456789	30Kč	2500
Oříšková čokoláda	Milka	+420123456789	30Kč	2800
Tyčinka milkyway	Milka	+420123456789	10Kč	7000
Mléčná čokoláda	Orion	+420987654321	25Kč	5800
Oříšková horalka	Horalka	+420897654321	7Kč	4560

Výrobek

Název	Výrobce_ID	Cena	Množství
Mléčná čokoláda	1	30Kč	2500
Oříšková čokoláda	1	30Kč	2800
Tyčinka milkyway	1	10Kč	7000
Mléčná čokoláda	2	25Kč	5800
Oříšková horalka	3	7Kč	4560

Výrobce

Vyrobce_ID	Vyrobce	Telefon
1	Milka	+420123456789
2	Orion	+420987654321
3	Horalka	+420897654321

3. normálna forma (3. NF)

- Relačná schéma je v 3. NF, keď je v 2. NF a každý atribút, kt. nie je primárny, je netranzitívne závislý na celom kľúči
- Schéma v 3NF môže byť redundantná

Zaměstnanec

r.č	Jméno	Příjmení	Město	PSČ	Funkce	Plat
1	Jack	Smith	Jihlava	58601	CEO	150000
2	Franta	Vomáčka	Praha10	10000	Senior Software Architect	80000
3	Pepa	František	Praha10	10000	Senior Software Architect	80000
4	Pavel	Novák	Kocourkov	99999	Junior Developer	30000
5	Petr	Koukal	Plzeň	12345	Database Designer	75000
6	Honza	Novák	Plzeň	12345	Junior Developer	30000

Zaměstnanec

r.č	Jméno	Příjmení	Město_ID	Funkce_ID
1	Jack	Smith	1	1
2	Franta	Vomáčka	2	2
3	Pepa	František	4	2
4	Pavel	Novák	3	4
5	Petr	Koukal	2	3
6	Honza	Novák	4	4

Město

Město_ID	Město	PSČ
1	Jihlava	58601
2	Praha10	10000
3	Kocourkov	99999
4	Plzeň	12345

Funkce

Funkce_ID	Funkce	Plat
1	CEO	150000
2	Senior Software Architect	80000
3	Database Designer	75000
4	Junior Developer	30000

- Závislosť *r.č* $\rightarrow$ *Město* $\rightarrow$ *PSČ* je tranzitívna závislosť *PSČ* na kľúči, rovnako ako závislosť *r.č.* $\rightarrow$ *Funkce* $\rightarrow$ *Plat*
- *Zaměstnanec* sa rozloží na *Zaměstnanec*, *Město* a *Funkce*

Boyce-Coddova normálna forma (BCNF)

- Relačná schéma je v Boyce-Coddovej normálnej forme, ak je v 3. NF a každý atribút je triviálne závislý na kľúči (teda každá závislosť v relačnej schéme je závislosť na kľúči)
- Jednoducho povedané, nie sú tam hodnoty `null`
- Zdroje: `http://www.manualy.net/article.php?articleID=13`

PB007 Softwarové inženýrství I

Cvičenie 9 – Design Class Diagram

Valdemar Švábenský

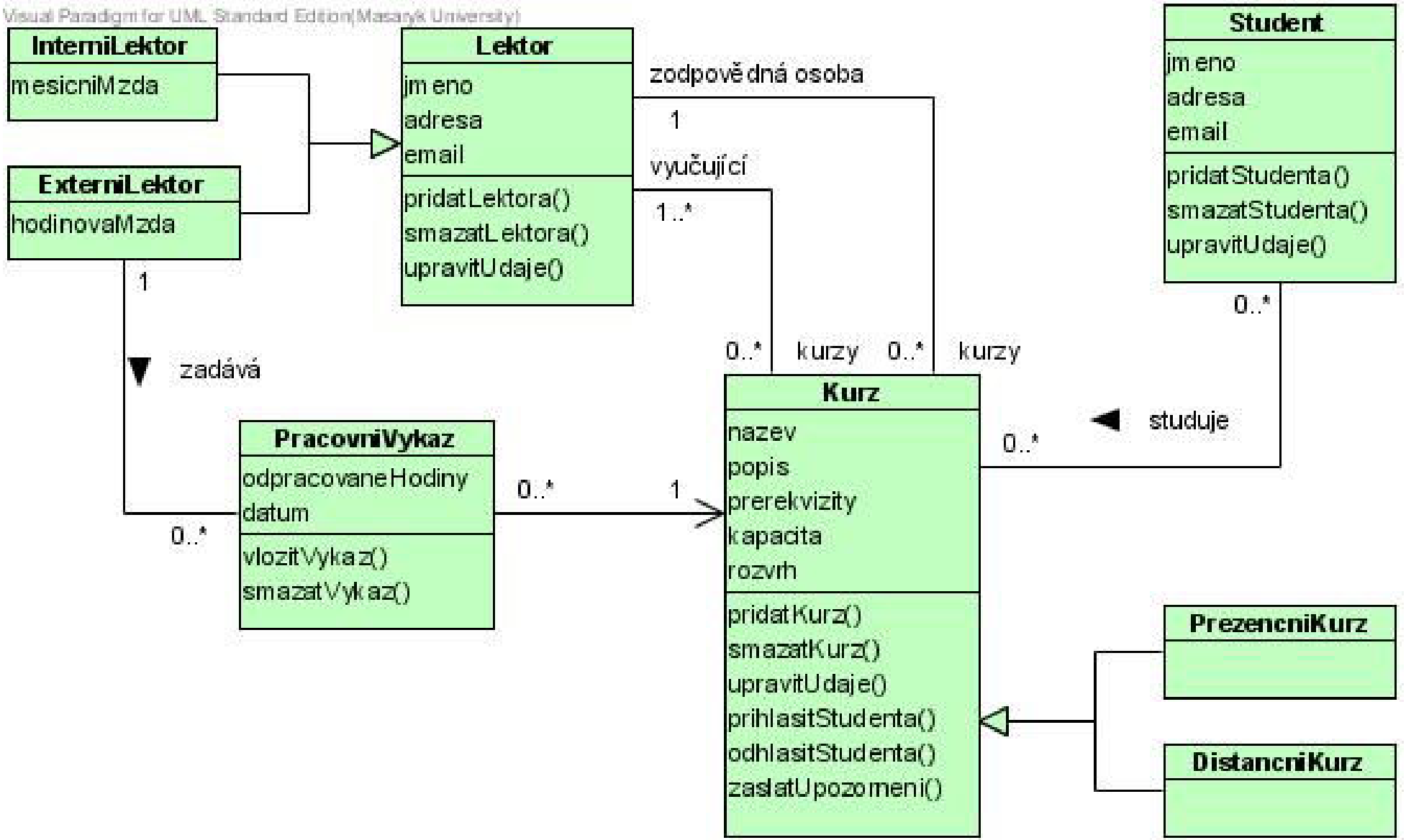
Fakulta informatiky, Masarykova univerzita, Brno

10. novembra 2015

Diagram tried – základný popis

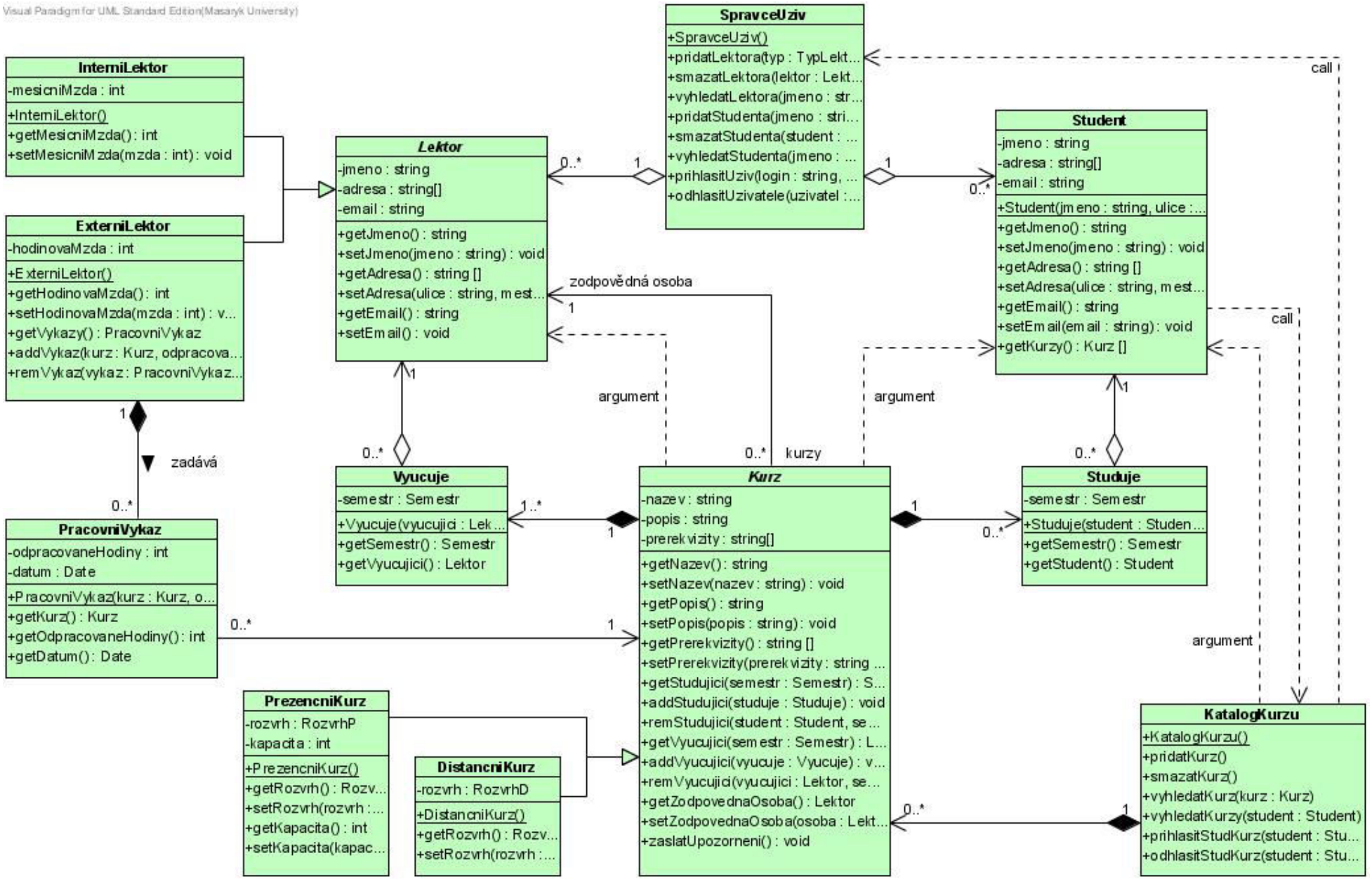
- Diagram tried modeluje statickú štruktúru objektovo orientovaného systému
- **Analytický diagram tried** jednoducho a prehľadne modeluje triedy a vzťahy medzi nimi
- **Návrhový diagram tried** vzniká pridaním implementačných detailov do analytického diagramu tried

Analytický diagram tried – ukážka



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/06/06_Studium_ClassAnalyza.jpg

Návrhový diagram tried – ukážka



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/10/10_Studium_ClassNavrh.jpg

Návrhový diagram tried

- Popisuje atribúty a metódy tried tak, že je možné podľa neho programovať (v konkrétnom jazyku)
  - Atribúty: viditeľnosť, typ, východzie hodnoty
  - Metódy: viditeľnosť, argumenty, typ návratovej hodnoty
- Do analytického diagramu sa pridávajú:
  - Metódy, ktoré vznikli rozložením analytických operácií, konštruktory, get/set metódy, ...
  - Triedy vyžadované použitou technológiou – pre prácu s GUI, databázou, ...
- Pre 1 analytickú triedu môže existovať viac návrhových

Vzťahy typu celok – časť

Agregácia:

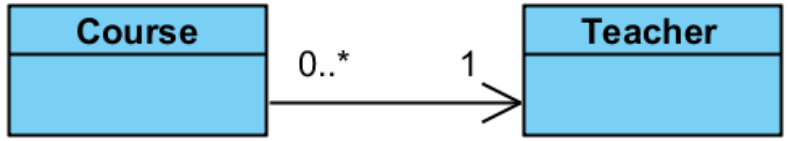
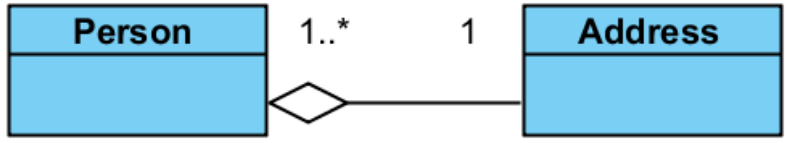
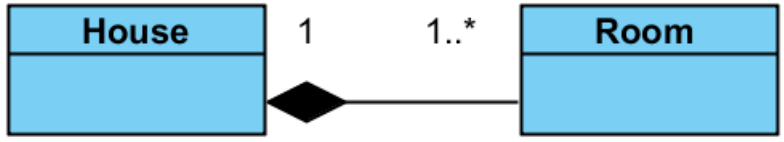
- Celok môže/nemusí existovať bez časti
- Časť môže existovať bez celku
- Časť môže byť zdieľaná viacerými celkami

Kompozícia:

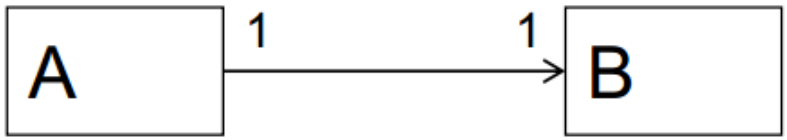
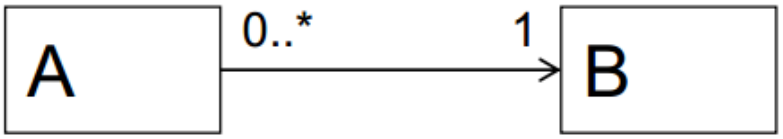
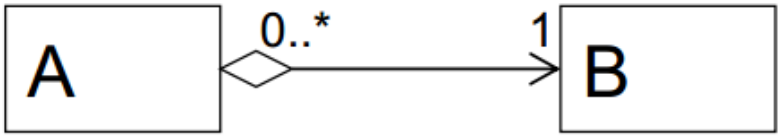
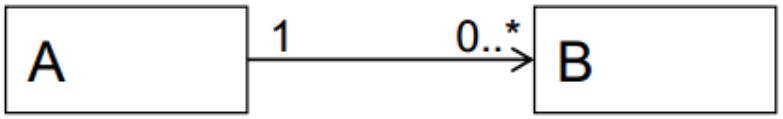
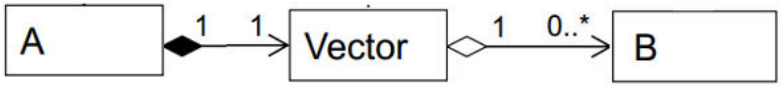
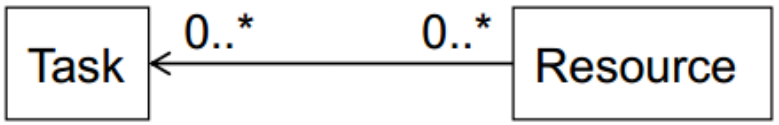
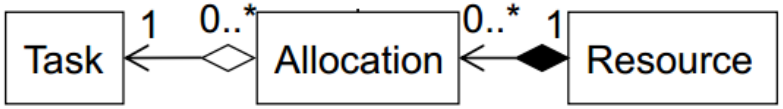
- Celok zodpovedá za svoje časti (vytvára a maže ich)
- Časť nemôže existovať samostatne (bez celku)
- Časť patrí vždy práve jednému celku
- Ak je celok zrušený, musí buď zrušiť všetky svoje časti, alebo ich presunúť k inému objektu

Vzťahy sú tranzitívne a asymetrické. Nesmú sa vyskytnúť cykly.

Vzťahy medzi triedami

	Popis	Príklad
Asociácia	obojsmerné prepojenie medzi triedami	
Agregácia	druh asociácie vyjadrujúci vzťah celok – časť	
Kompozícia	silnejšia forma agregácie	

Rozpracovanie asociácií

	Analýza	Návrh
1:1		
M:1		
1:N		
M:N		

Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/06-Design.pdf

PB007 Softwarové inženýrství I

Cvičenia 10 a 11 – Interaction Diagrams

Valdemar Švábenský

Fakulta informatiky, Masarykova univerzita, Brno

24. novembra 2015

Interakčný diagram

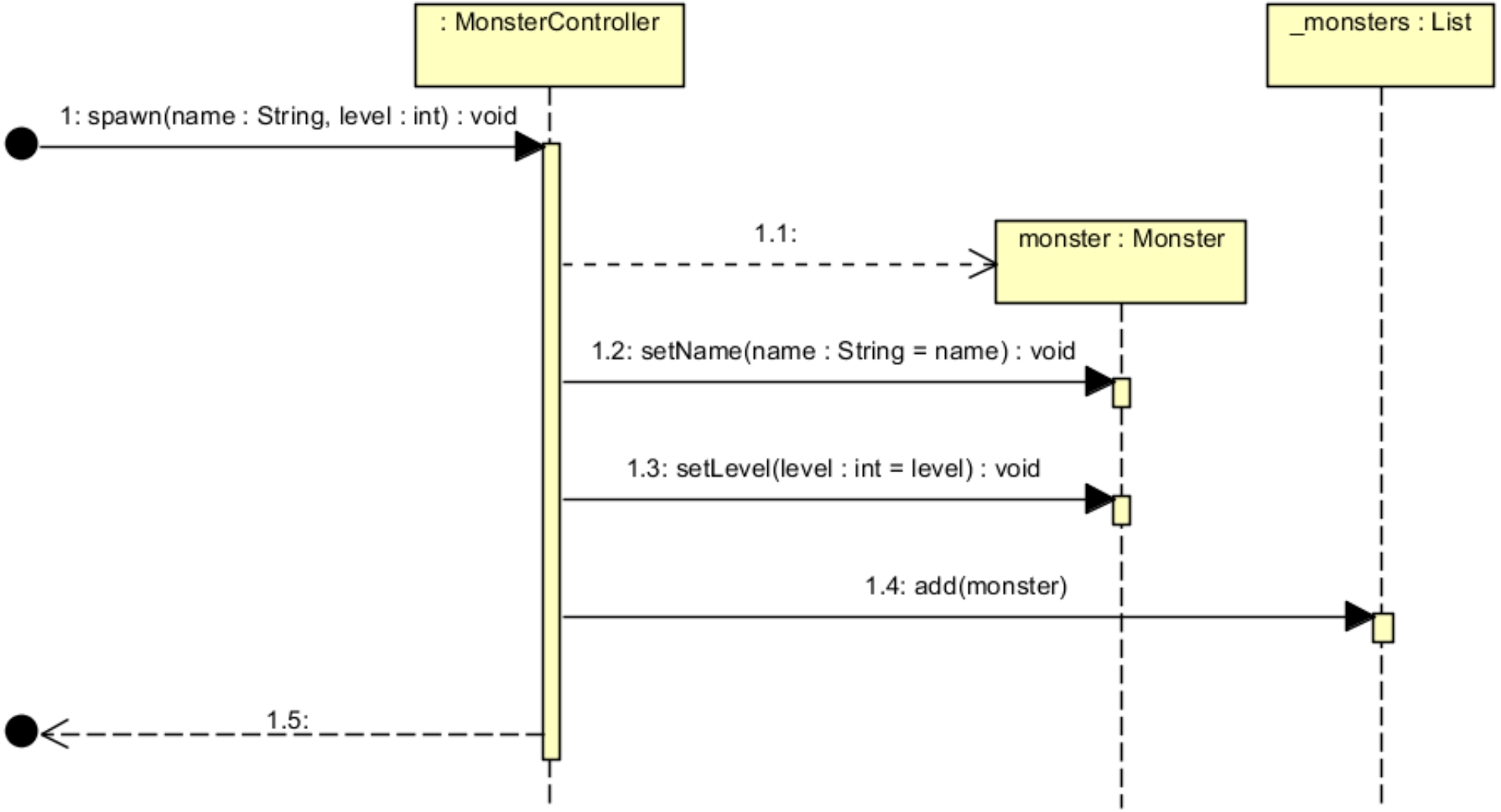
- Popisuje spoluprácu medzi skupinami objektov systému (triedy, prípady použitia, . . .)
- Viac typov, najčastejšie sa používa:
  1. Sekvenčný diagram (Sequence Diagram) – dôraz na časovú postupnosť zasielania správ
  2. Komunikačný diagram (Communication Diagram) – dôraz na vzťahy medzi objektami („kto s kým komunikuje“)

Sekvenčný diagram – intuícia

- Majme triedu `Monster` s atribútmi `name` a `level` a verejnými metódami `setName()` a `setLevel()`
- Uvážte ďalšiu triedu `MonsterController`, ktorá zodpovedá za uchovávanie všetkých aktuálnych príšer:

```
public class MonsterController {
  private List _monsters = new List();
  public void spawn(String name, int
      level) {
    Monster monster = new Monster();
    monster.setName(name);
    monster.setLevel(level);
    _monsters.add(monster);
  }
}
```

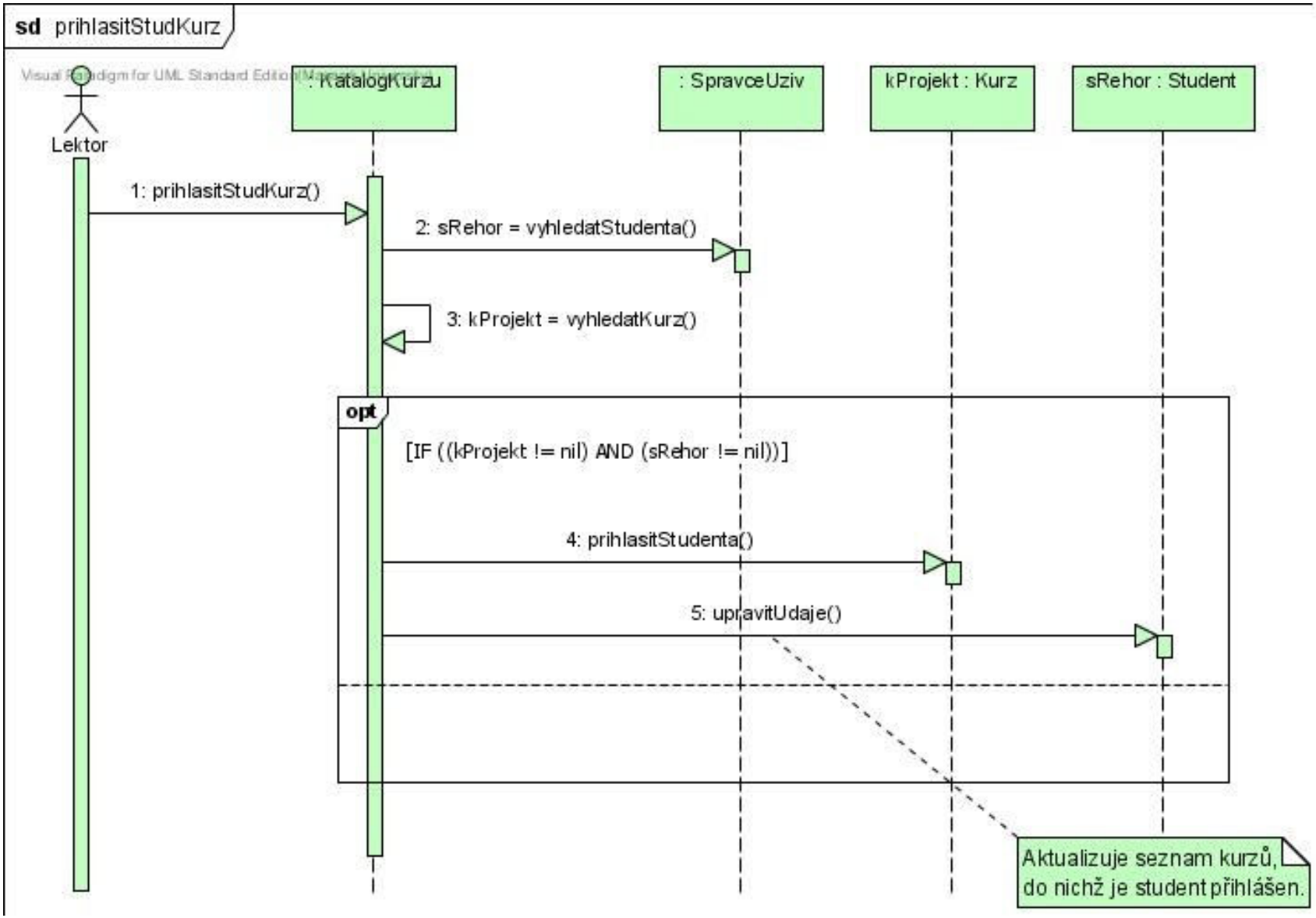
Sekvenčný diagram – intuícia



Sekvenčný diagram

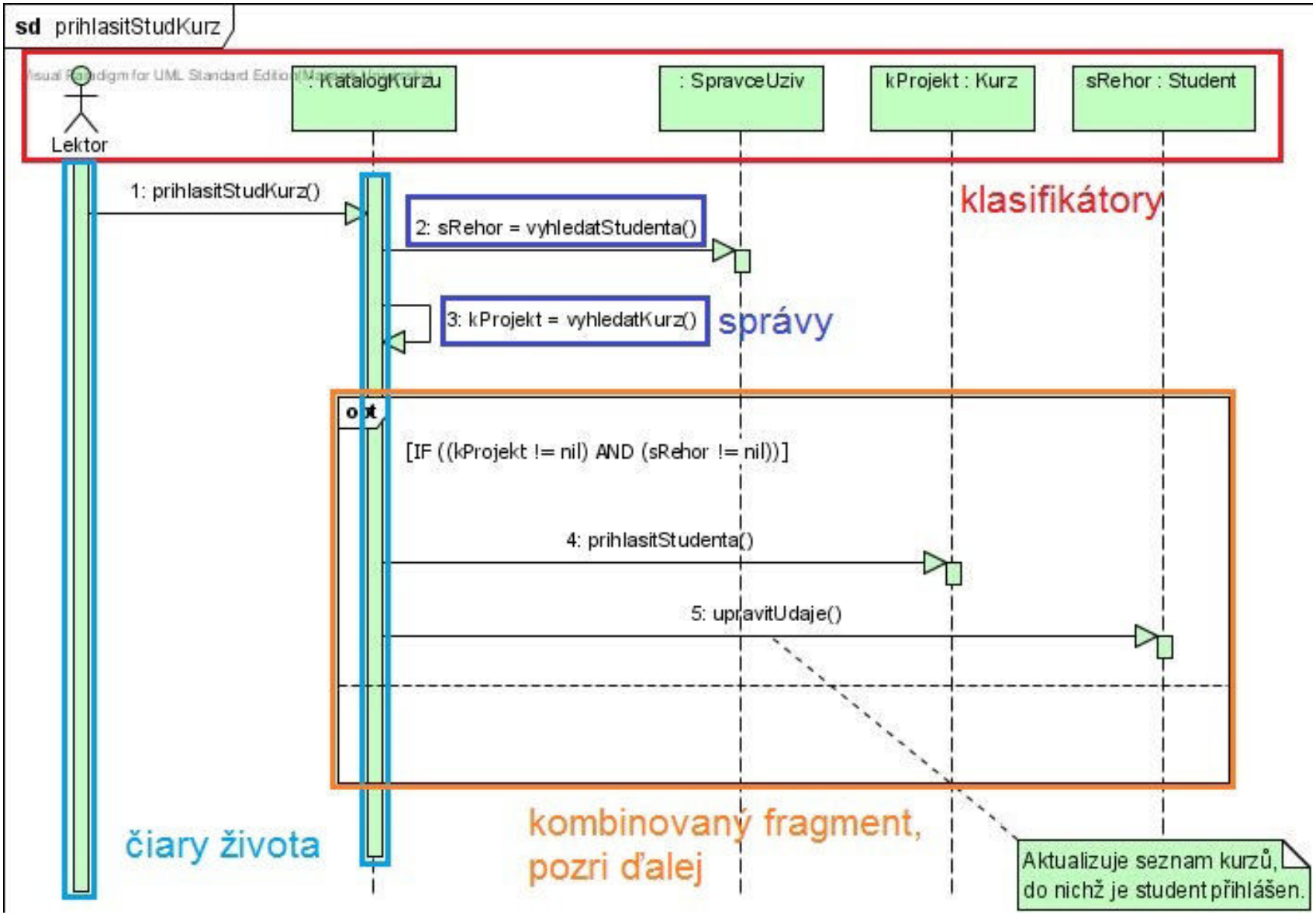
- Zachytáva komunikáciu medzi objektami s dôrazom na časovú postupnosť zasielania správ
- Správy sú vymieňané medzi tzv. **klasifikátormi** (classifiers) – aktér, trieda alebo objekt
- Z každého klasifikátora vedie zvislá **čiara života** (lifeline) – reprezentuje jeho život počas komunikácie
- **Správa** (message) je požiadavka objektu o vyvolanie operácie druhého objektu
  - Šípka medzi čiarami života
  - Objekty môžu zasielať správy aj samy sebe
  - Poradie zasielania správ určuje v diagrame os Y
  - Čas plynie zhora dole

Sekvenčný diagram – ukážka



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/09/09_Sequence_prihlasitStudKurz-1.jpg

Sekvenčný diagram – ukážka

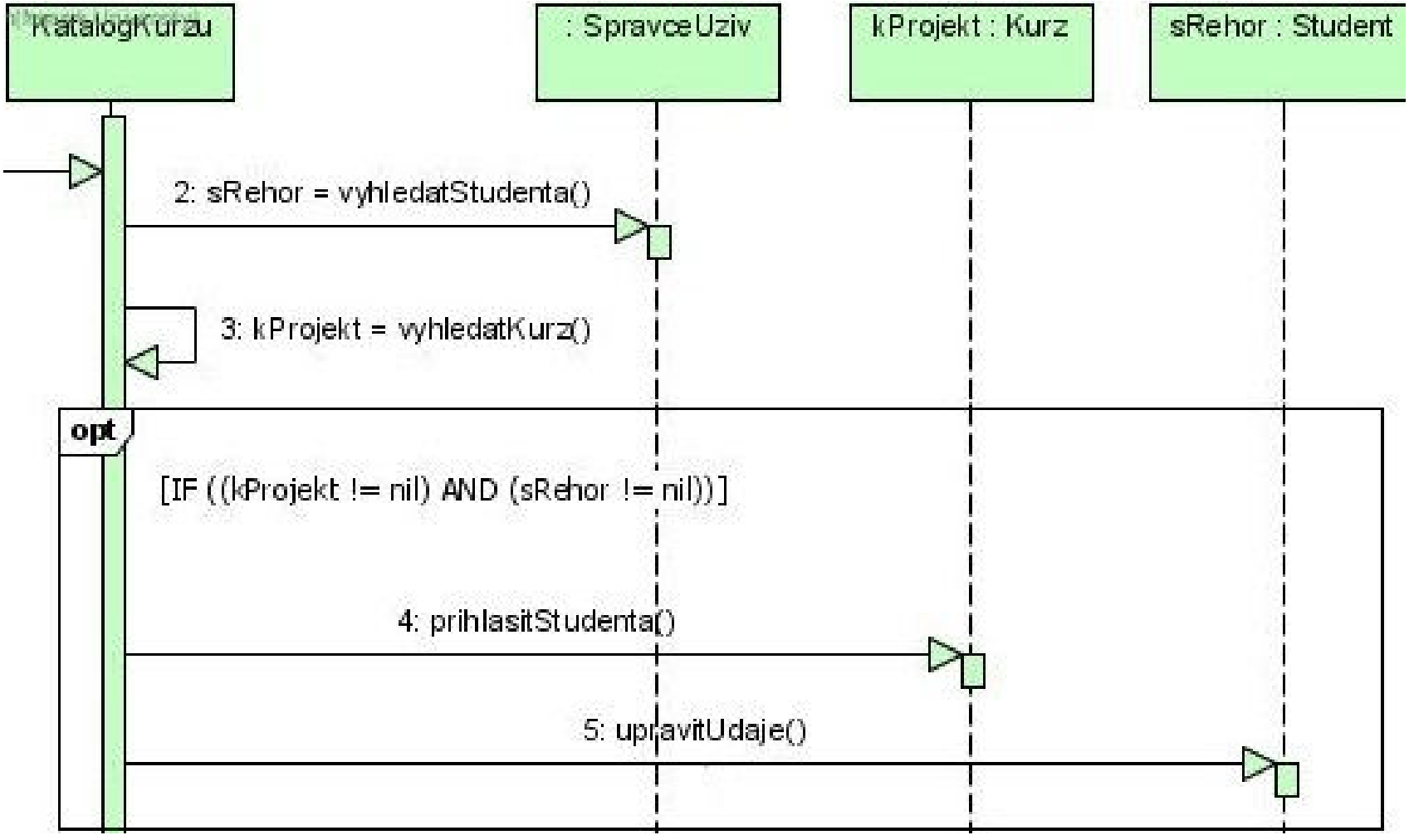


Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/09/09_Sequence_prihlasitStudKurz-1.jpg

Sekvenčný diagram – kombinovaný fragment

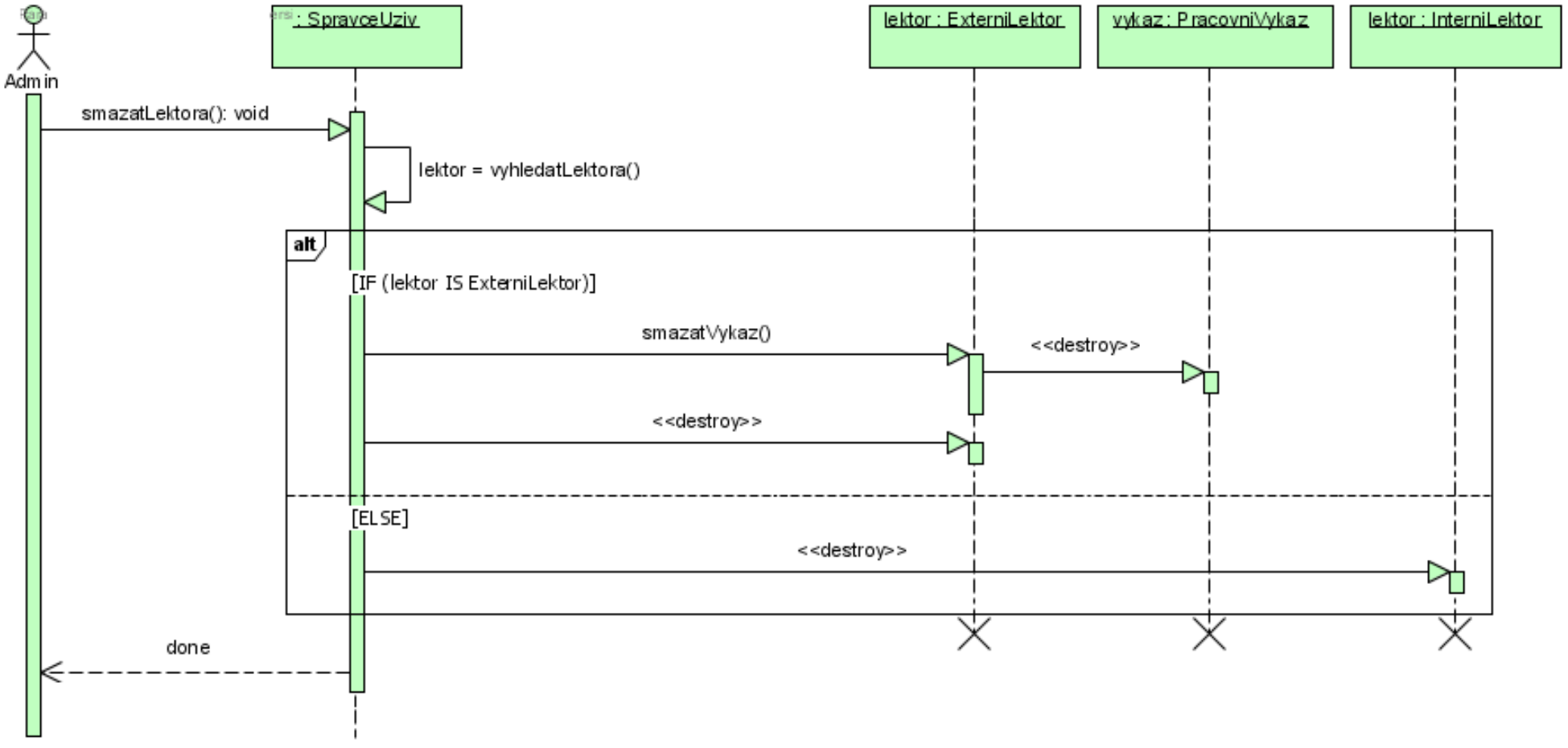
- Oblasť vnútri sekvenčného diagramu s odlišným chovaním
- Každý kombinovaný fragment tvorí:
  - 1 operátor
  - 1 alebo viac operandov (skupín správ v rámci fragmentu)
  - 0 alebo viac podmienok
- Operátory:
  - `opt` (option) – má 1 operand, ktorý sa spustí, len ak je splnená definovaná podmienka
  - `alt` (alternatives) – má aspoň 2 operandy; spustí sa ten, ktorého podmienka sa vyhodnotí na `true`
  - `loop` – opakované vykonávanie 1 operandu
  - `break` – má 1 operand, ktorý sa spustí v prípade splnenia podmienky a ukončí vykonávanie cyklu

Sekvenčný diagram – opt



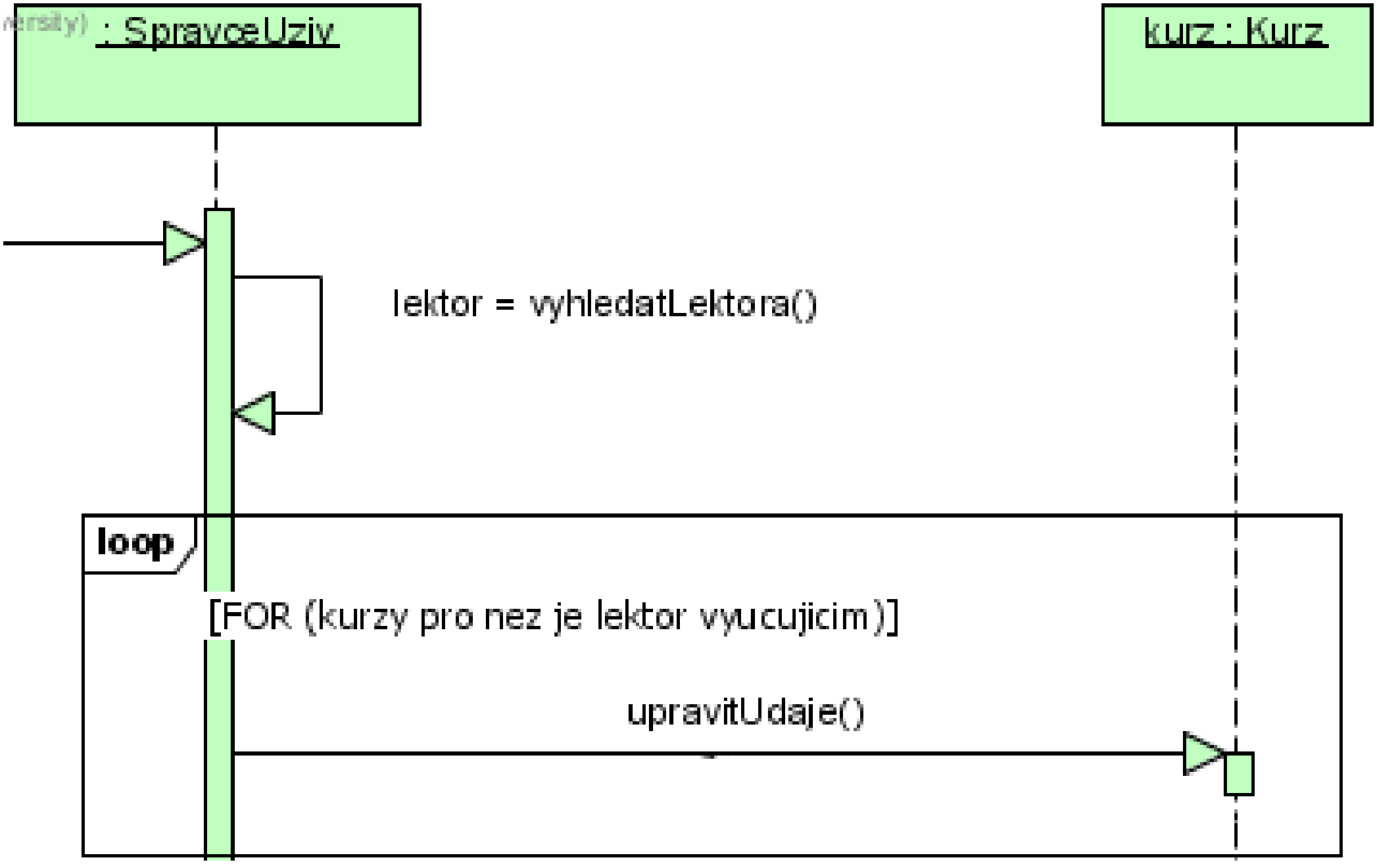
Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/09/09_Sequence_prihlasitStudKurz-1.jpg

Sekvenčný diagram – alt



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/08/08_Sequence_odstranitLektora.png

Sekvenčný diagram – `loop`

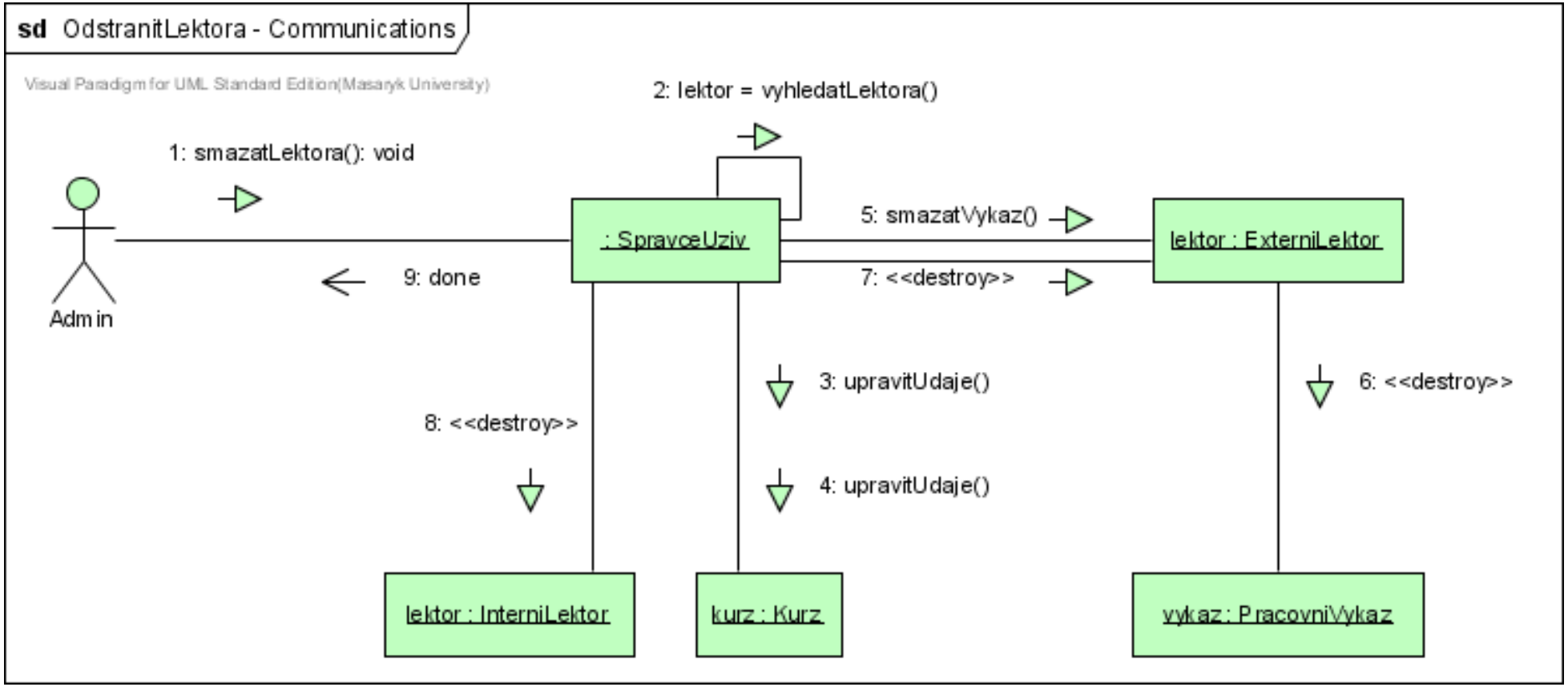


Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/08/08_Sequence_odstranitLektora.png

Komunikačný diagram

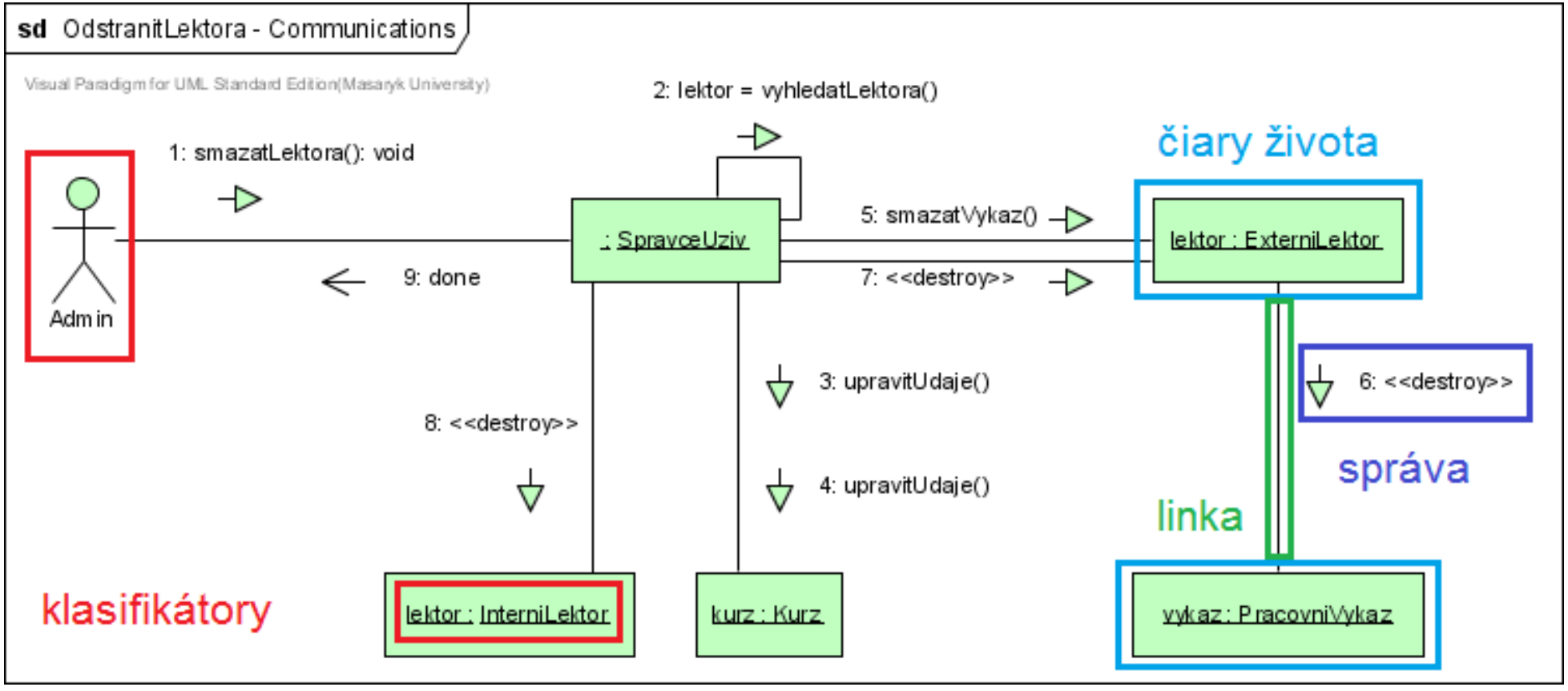
- Zachytáva komunikáciu medzi **klasifikátormi** s dôrazom na vzťahy medzi nimi
- **Čiara života** reprezentuje klasifikátor (ako obdĺžnik)
- **Správa** – ako v sekvenčnom diagrame, ale musí byť číslovaná
- Správy putujú po **linkách** (link) – obojsmerný spoj medzi dvoma čiarami života

Komunikačný diagram – ukážka



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/08/08_Comm_odstranitLektora.png

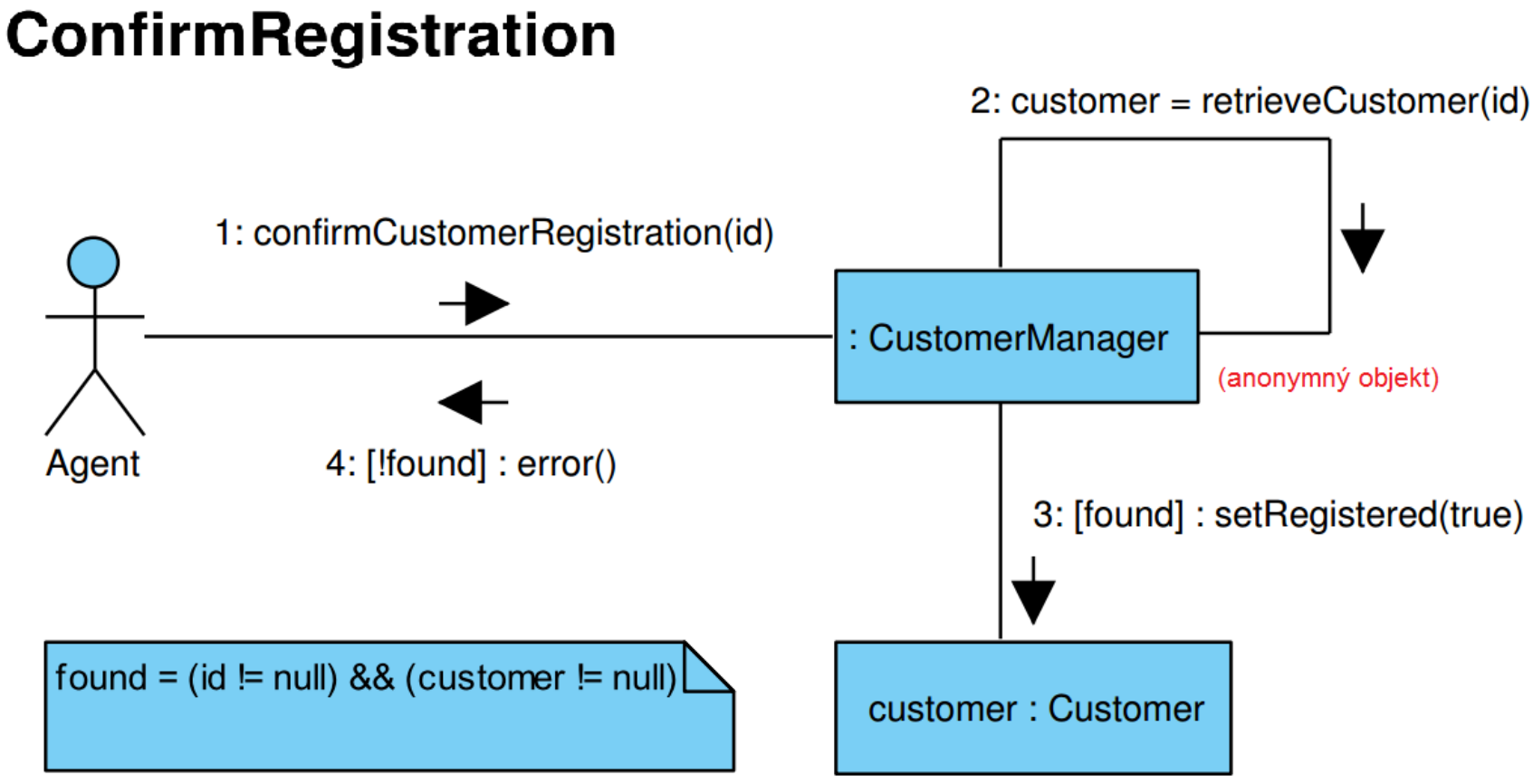
Komunikačný diagram – ukážka



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/08/08_Comm_odstranitLektora.png

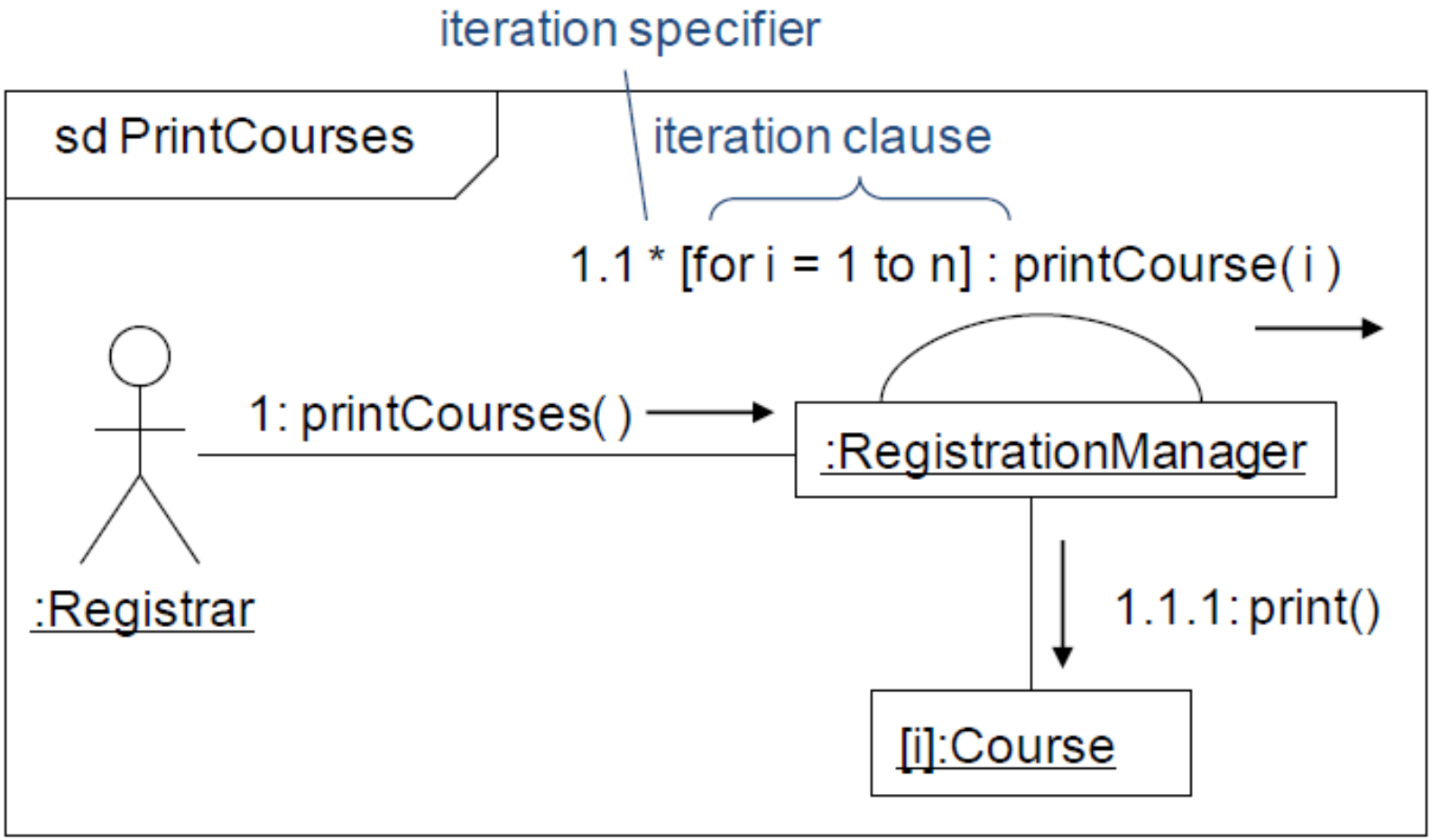
Komunikačný diagram – vetvenie

- Komunikačný diagram je syntakticky slabší – neobsahuje konštrukcie pre tvorbu podmienok a cyklov
- Môžete si dopomôcť pseudokódom



Komunikačný diagram – iterácia

- Komunikačný diagram je syntakticky slabší – neobsahuje konštrukcie pre tvorbu podmienok a cyklov
- Iteračný výraz: `* [loop min, max [condition]]`



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/07-Implementation.pdf

Tipy pre modelovanie

- Názvy klasifikátorov zodpovedajú názvom objektov v iných diagramoch (Use Case Diagram, Class Diagram)
- Názvy správ zodpovedajú názvom metód v diagrame tried
- $\Rightarrow$ **Modelujte podľa diagramu tried!**

- Klasifikátory sú radené podľa poradia zasielania správ
- Správy sú číslované podľa poradia
- U komunikačných diagramov má každá správa smer

PB007 Softwarové inženýrství I

Cvičenie 12 – Package, Component, Deployment Diagrams

Valdemar Švábenský

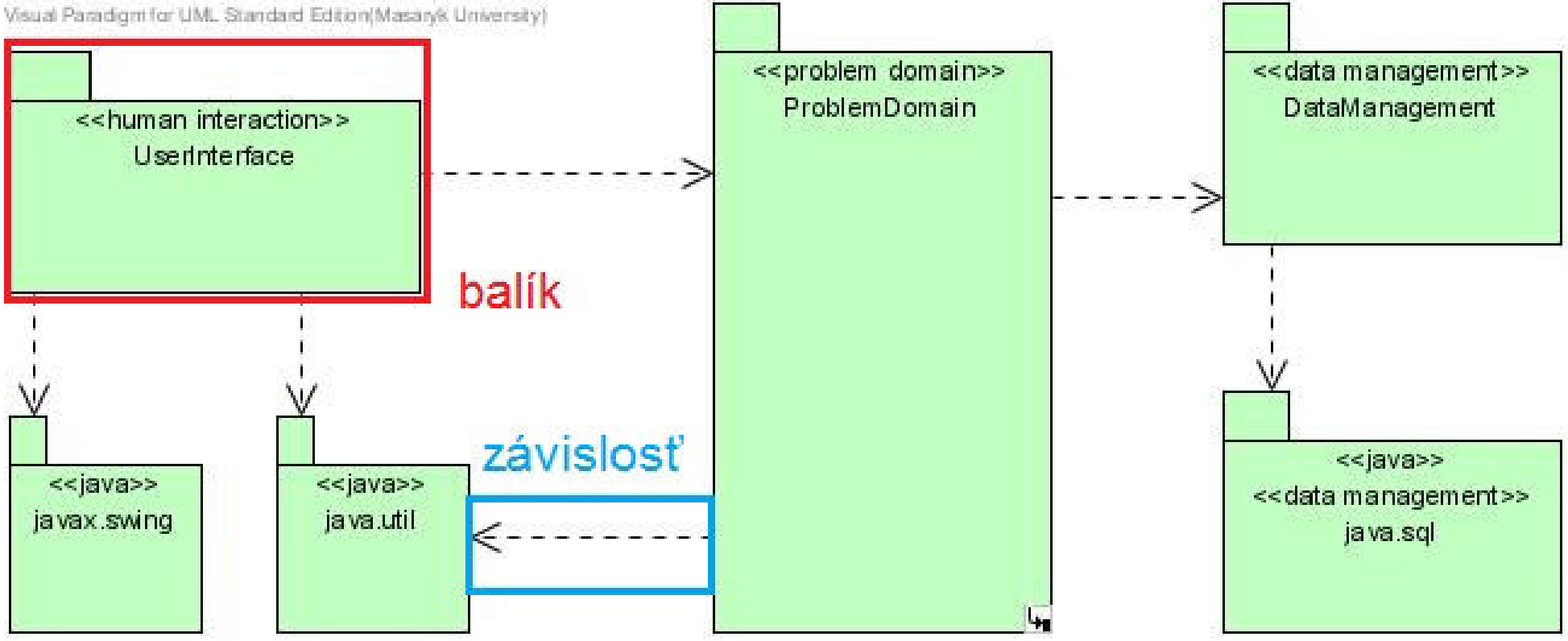
Fakulta informatiky, Masarykova univerzita, Brno

8. decembra 2015

Diagram balíkov

- Návrhy veľkých systémov obsahujú stovky tried
- Diagram balíkov logicky združuje prvky rôznych diagramov do skupín a popisuje závislosti medzi nimi
- Analógia s balíkom (package) v Jave
- Prvky diagramu:
  - **Balík** (Package) – mechanizmus pre združovanie sémanticky súvisiacich prvkov v diagramoch
  - **Závislosť** (Dependency) – relácia medzi balíkmi, kedy je klientský balík nejako závislý na zdrojovom balíku
    - «use» – Klientský balík využíva prvky zdrojového balíku (východzí typ závislosti)
    - «import» – Verejné prvky zdrojového balíka sú pridané ako verejné prvky do klientského balíku
    - «access» – Verejné prvky zdrojového balíka sú pridané ako súkromné prvky do klientského balíku

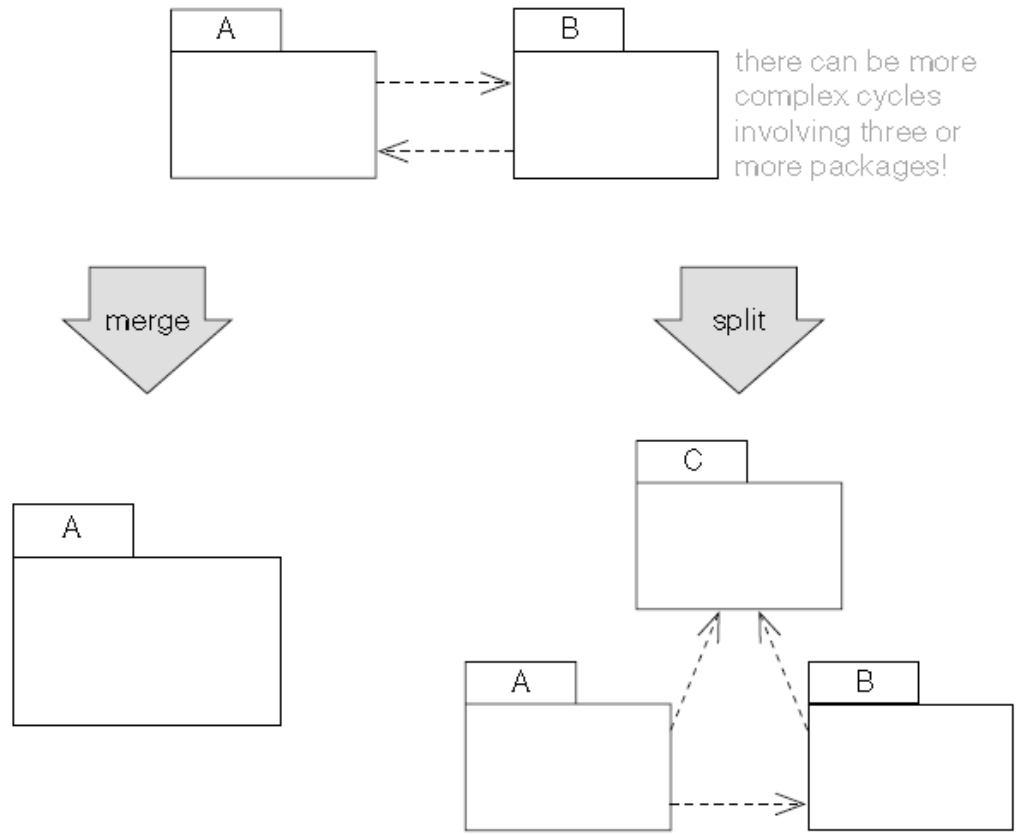
Diagram balíkov – ukážka



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/11/11_StudiumPackage.jpg

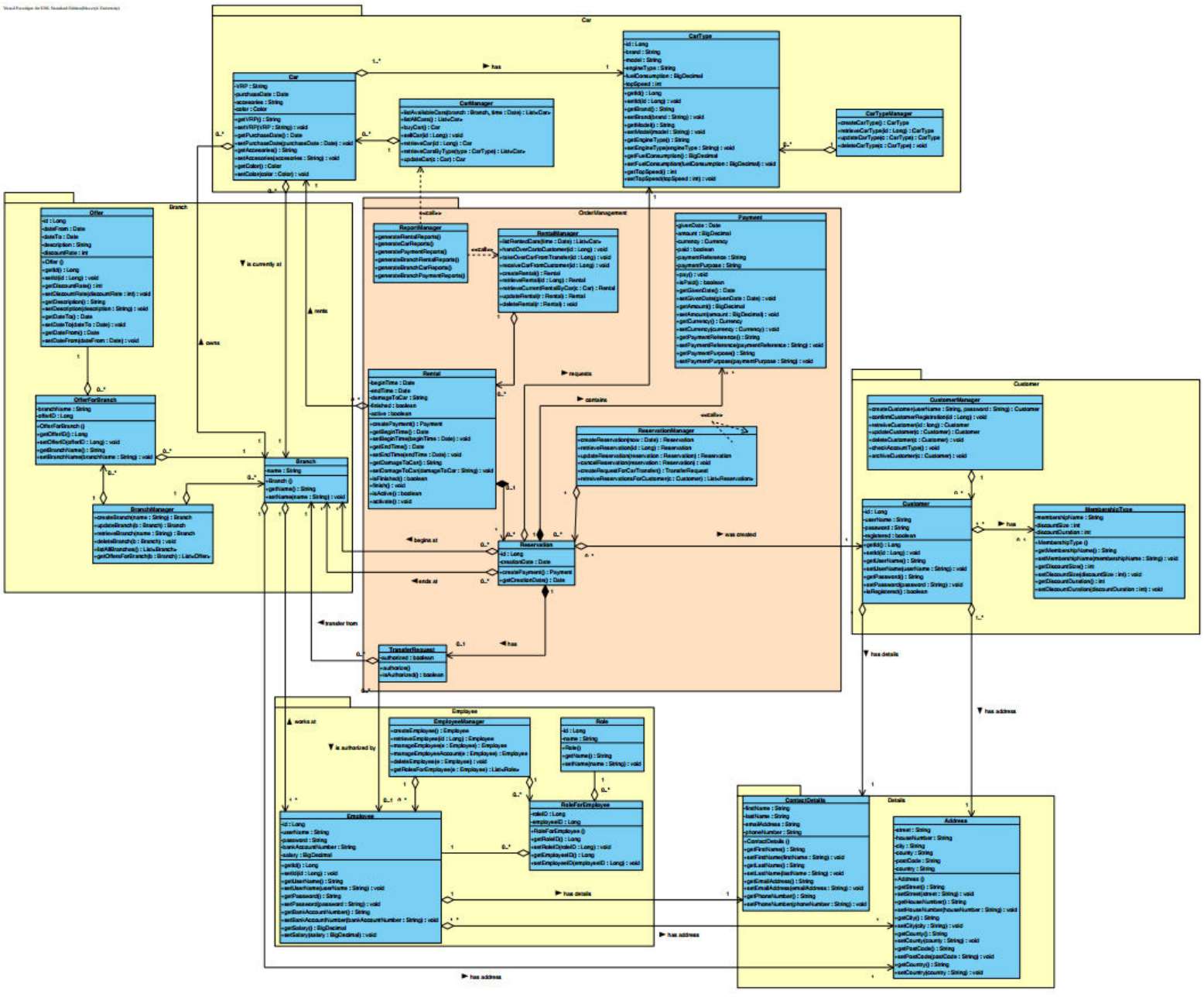
Diagram balíkov – závislosti

- Závislosť medzi balíkmi $\Longleftrightarrow$ existuje závislosť medzi dvoma prvkami týchto balíkov
- Závislosť je tranzitívna
- Nesmie dochádzať k cyklickým závislostiam:



Zdroj: https://praveenthomasln.wordpress.com/tag/uml-packages/

Diagram balíkov a tried (ilustračný obrázok)



Výsledný samostatný diagram balíkov

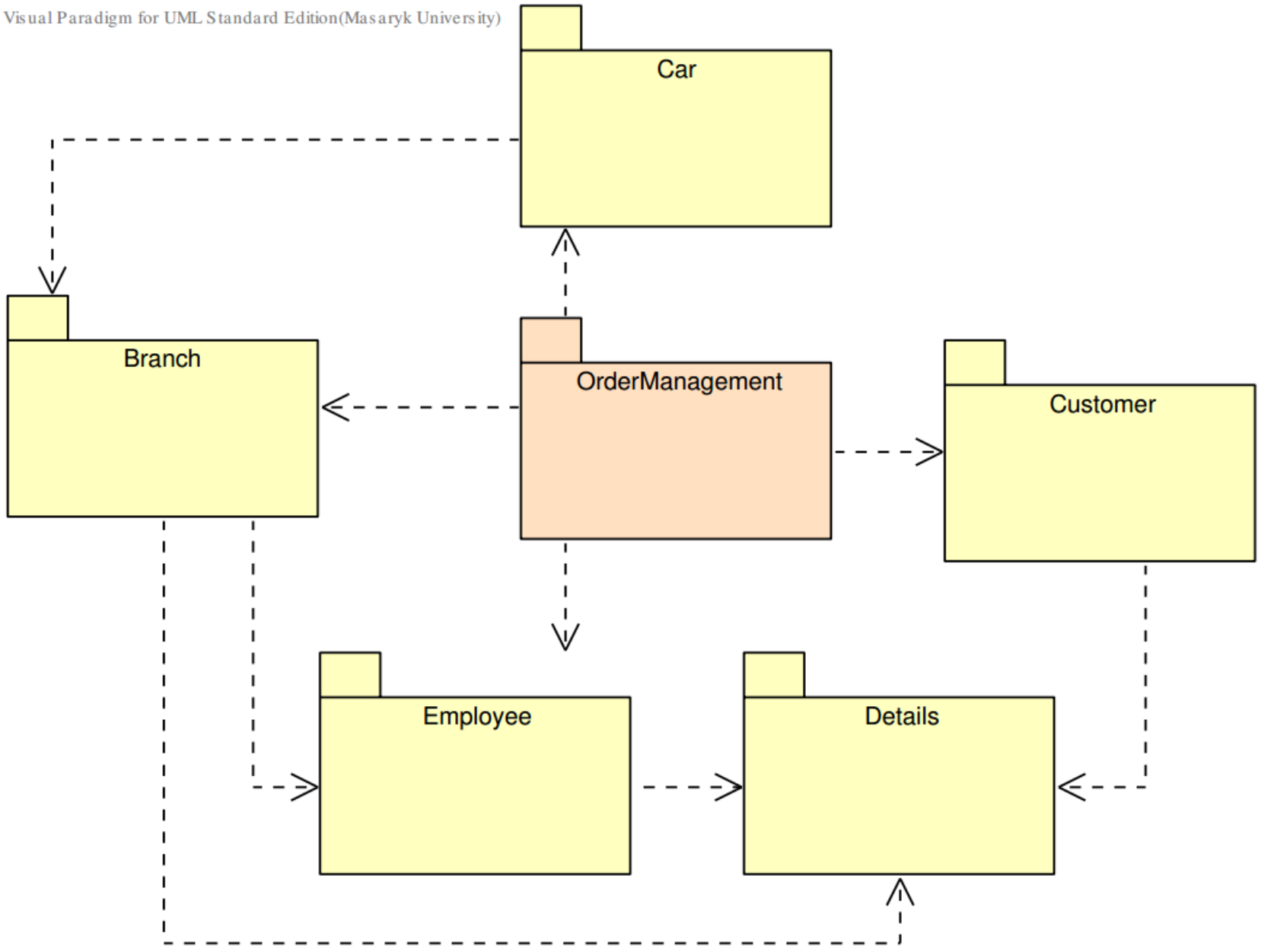
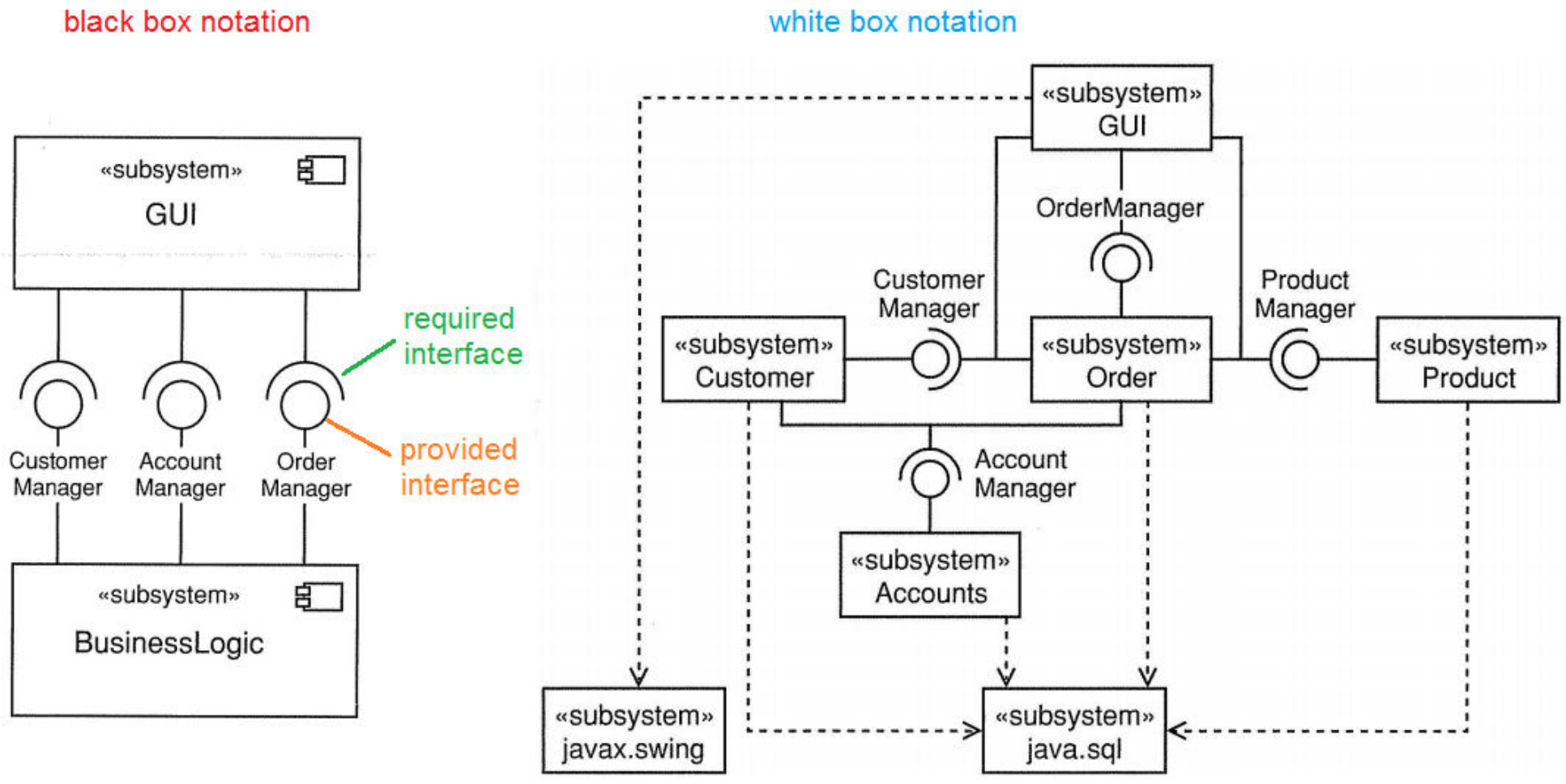


Diagram komponentov

- Komponent = modulárna časť systému, ktorej externé chovanie je úplne definované **poskytovanými** a **požadovanými rozhraniami**
  - Modulárna = môže byť nahradená iným komponentom, ak podporuje rovnaký protokol
- Black-box pohľad: vnútro komponentu je skryté
- White-box pohľad: vnútro komponentu je rozpracované

Diagram komponentov – ukážka

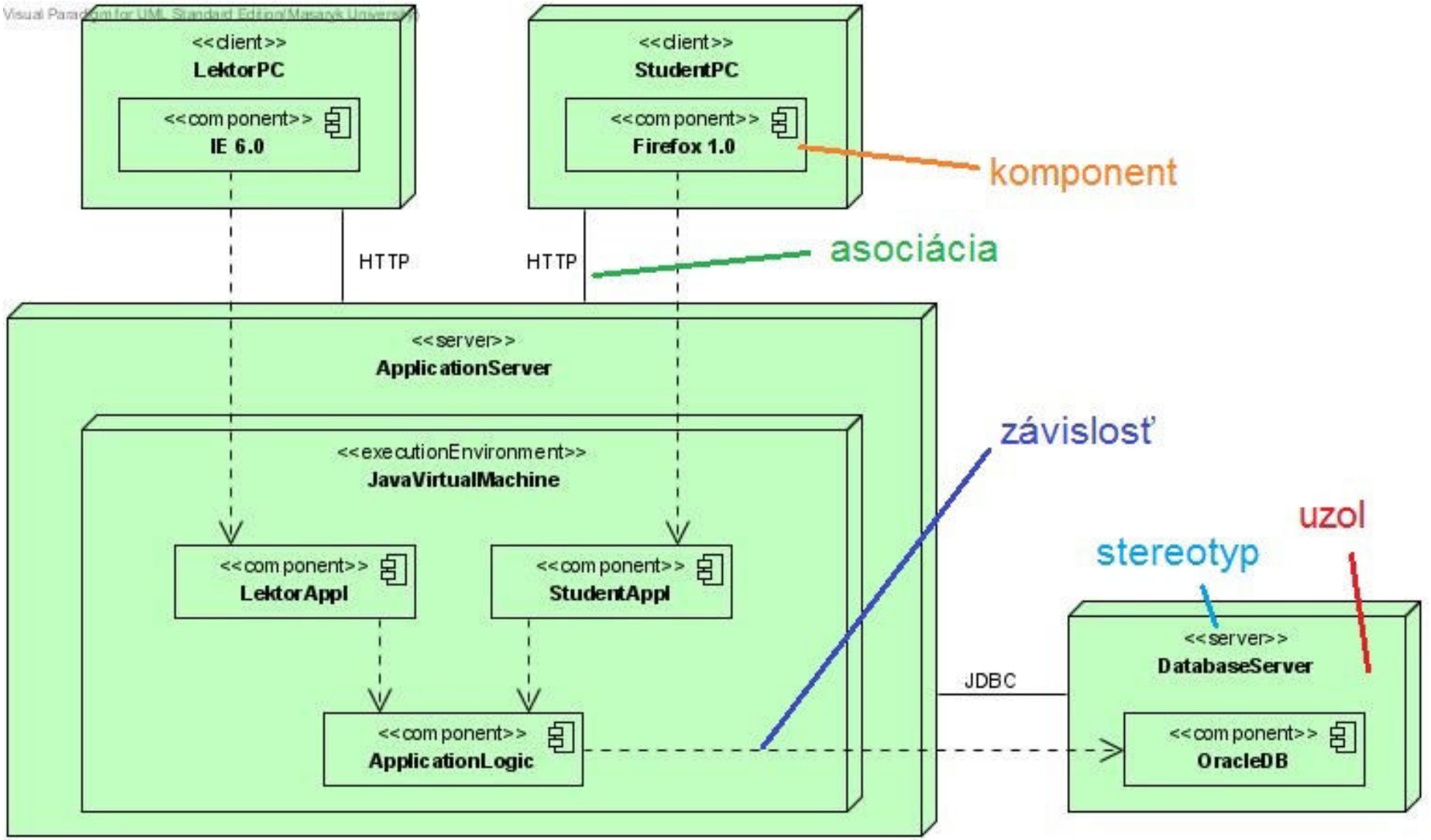


Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/lec/08-Architecture.pdf

Diagram nasadenia

- Diagram nasadenia zobrazuje, ako bude architektúra SW mapovaná na architektúru HW
- Prvky:
  - **Uzly** (nodes) – typ výpočtového zdroja, na ktorý budú umiestnené jednotlivé časti systému
    - «device» – zariadenie, HW
    - «execution environment» – SW prostredie
  - **Artefakty / Komponenty** – SW nasadený na uzloch
    - Artefakt – fyzická úroveň, napr. súbor
    - Komponent – logická úroveň, zoskupenie artefaktov
  - **Rozhrania** pre komunikáciu s komponentami
  - **Asociácie** – spojenia (komunikačné kanály) medzi uzlami (+ názov komunikačného protokolu)
  - **Závislosti** medzi artefaktmi / komponentmi

Diagram nasadenia – ukážka



Zdroj: https://is.muni.cz/auth/el/1433/podzim2015/PB007/um/sem/cz_files/12/12_StudiumDeployment.jpg

Diagram nasadenia – ukážka 2