Unifikovaný vizuální modelovací jazyk (UML)

Jazyk pro :

- **Vizualizaci,**
- **Konstrukci,**
- **Specifikaci,**
- **Dokumentaci**

Artefaktů SW systémů

Proč unifikovaný?

- **Sjednocení předchozí metody a notace**
- **Podporuje celý vývojový cyklus**
- **Podporuje různé aplikační oblasti**
- **Volnost v použití metodiky**
- **Nezávislý na cílovém prog. Jazyku a doméně**
- **Založený na potřebách a zkušenostech a potřebách komunity.**
- **Reflektuje současné trendy vývoje SW**

UML

- **Jazyk = syntaxe + sémantika**
  - **Sytaxe – pravidla, jak tvořit výrazy s elementů**
  - **Sématika – jak syntaktickým výrazům přiřadit vývznam**
- **Není metodika – nepředepisuje, jak postupovat při vývoji SW**

UML Historie-

Programovací jazyky

- **Simula – třídy, objekt, polymorfismus, dědění**
- **SmalTalk – virtuální stroj**
- **C++ - výjimky**
- **Eifel – garbage collection**

Objektové modelovaci jazyky a metodiky

- **Booch - OOAD**
- **Rambauch - OMT**
- **Jacobson – Objectory**

1994 – zahájení vývoje UML – Rational Software (Rumbaugh, Booch)

1995 – notace pro Unified Method 0.8, připojuje se

- Jacobson a model jednání, Rational Unified Process

1995 – UML 1.0 (zabudování připomínek, kooperace)

1996 1997 – UML 1.1 (preciznější sémantika, přijat jako standard OMG)

1999 – UML 1.3 (upřesněná verze 1.1)

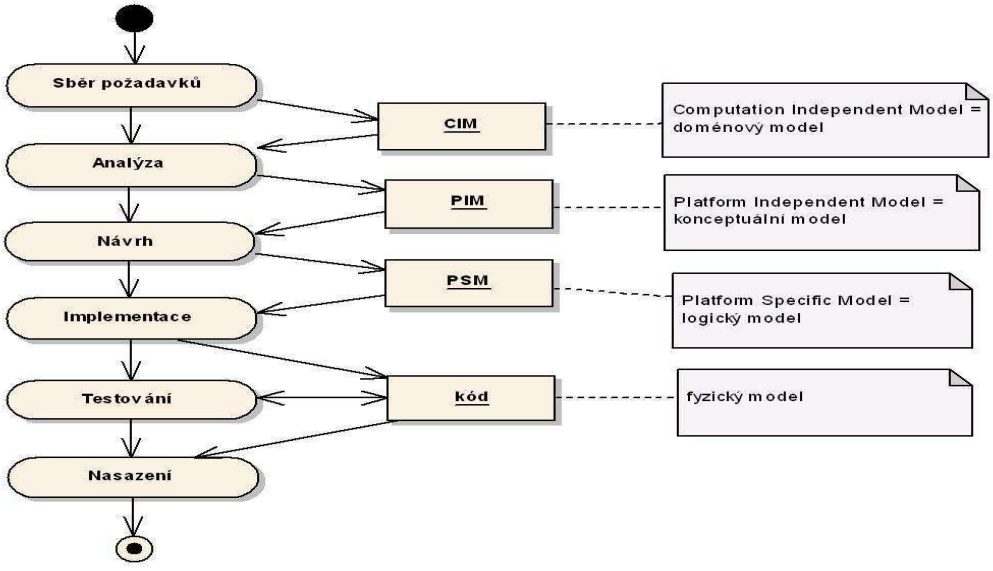
2001 – UML 1.4 (komponenty)

2003 – UML 1.5 (diagramy aktivit, sémantika akcí)

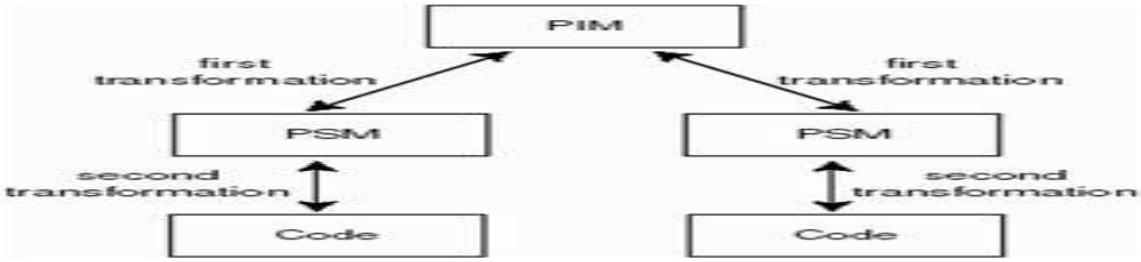
2005 – UML 2.0 (nové diagramy, změny)

2006 – UML 2.1 (další snaha o upřesnění sémantiky)

MDD – Model Driven Development



MDD – Model Driven Delelopment

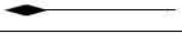
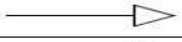


Objekty a UML
UML modeluje systém jako množinu spolupracujících objektů
Statiscká struktura
Jaké druhy objektů jsou důležité
Jaké jsou jejich vztahy
Dynamické chování
Životní cyklus objektu
Interakce objektů pro dosažení cílů
Struktura UML
Stavební bloky-obecné mechanizmy-architektura
Stavební bloky

- **Věci (Things)**
  - **Modelovací atomy**
    - **Vztahy**
  - **Pojí věci dohromady**
    - **Diagramy**
  - **Pohledy na významné spojení věcí**
  - **Různé pohledy na modelovaný systém**
    - **Věci**
  - **Strukturální abstrakce – podstatná jména modelu**
    - **Třídy, rozhraní, spolupráce, případy užití, aktivní třídy, komponenty, uzly**
  - **Absatrakce chování – slovesa modelu**
    - **Interakce, stavové automaty**
  - **Seskupení**
    - **Balíčky**
    - **Modely, rámce, podsystémy**
  - **Anotace**
    - **Poznámky**

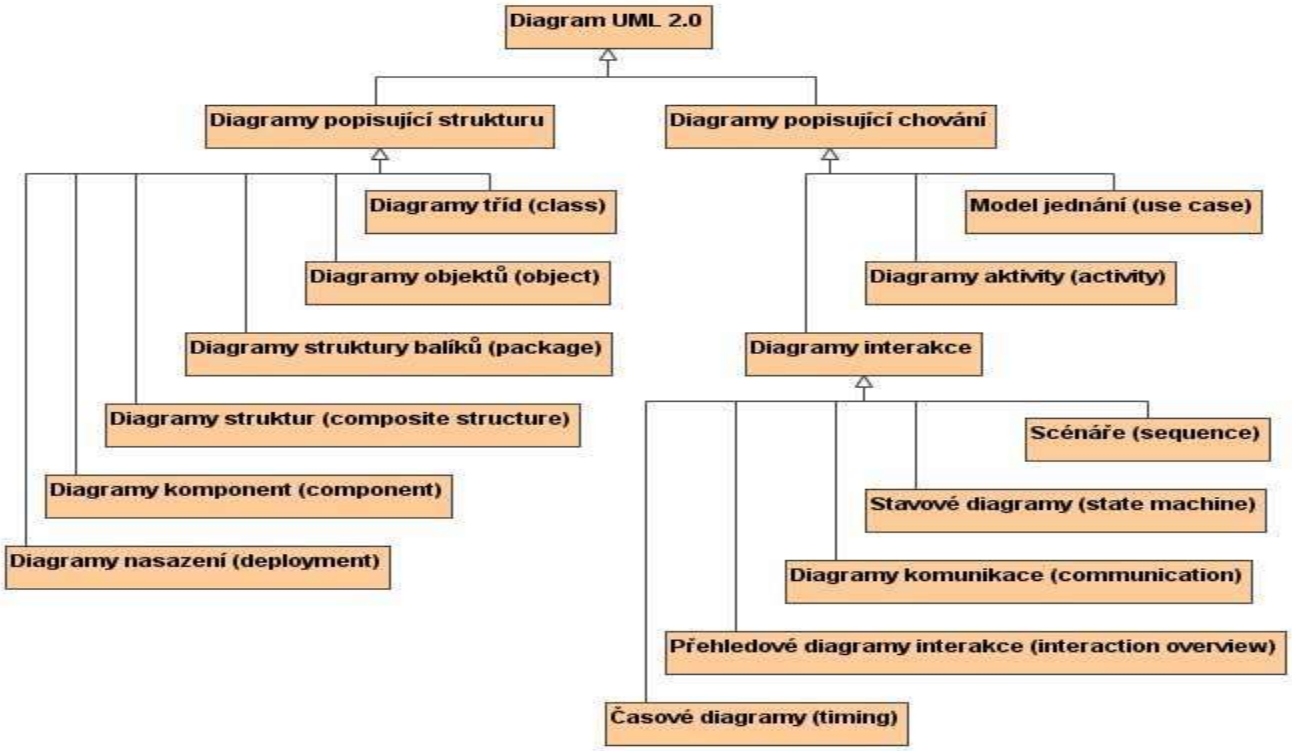
Vztahy

Vztahy

relationship	UML syntaxe	sémantika stručně
závislost		Zdrojový element závisí na cílovém.
asociace		Vazba mezi objekty.
agregace		Cílový element je částí zdrojového
kompozice		Silnější forma agregace.
obsahuje		Zdrojový element obsahuje cílový element.
generalizace		Zdrojový element je specializací cílového.
realizace		Zdrojový element zaručuje splnění kontraktu předepsaného cílovým element.

Úvod do UML 14

UML má 13 typů diagramů



Diagramy

Diagramy

UML 1.5	UML 2.0
Diagram tříd (class diagram)	Diagram tříd (class diagram)
Diagram objektů (object diagram)	Diagram objektů (object diagram)
Model jednání (use case diagram)	Model jednání (use case diagram)
Scénář (sequence diagram)	Scénář (sequence diagram)
Diagram kolaborace (collaboration diagram)	Diagram komunikace (communication diagram)
Diagram aktivity (activity diagram)	Diagram aktivity (activity diagram)
Stavový diagram (statecharts diagram)	Stavový diagram (state machine diagram)
Diagram komponent (commponent diagram)	Diagram komponent (commponent diagram)
Diagram nasazení (deployment diagram)	Diagram nasazení (deployment diagram)
	Diagram struktury (composite structure diagram)
	Diagram struktury balíků (package diagram)
	Přehledový diagram interakce (interaction overview diag.)
	Časový diagram (timing diagram)

Úvod do UML 16

Syntaxe diagramu

Syntaxe diagramu

syntaxe hlavičky: <druh> <jméno> <parametry>
<druh> a <parametry> jsou volitelné

Diagram tříd

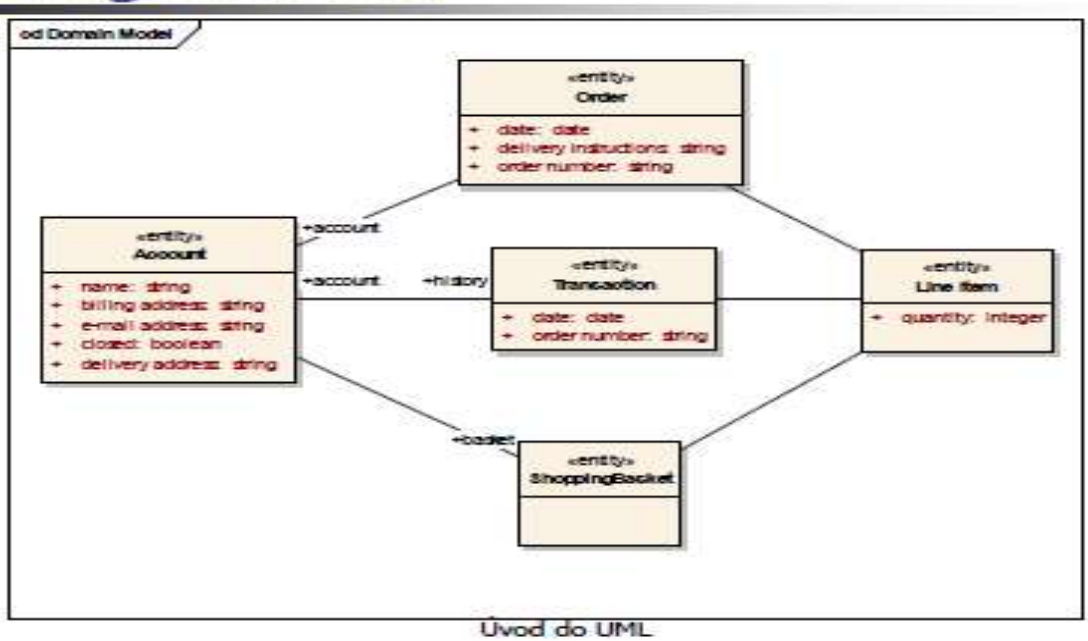


Diagram případů užití

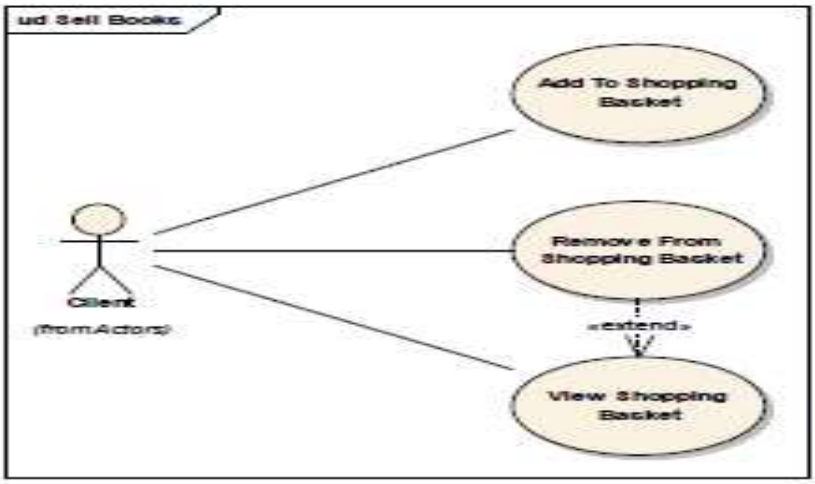


Diagram komponent

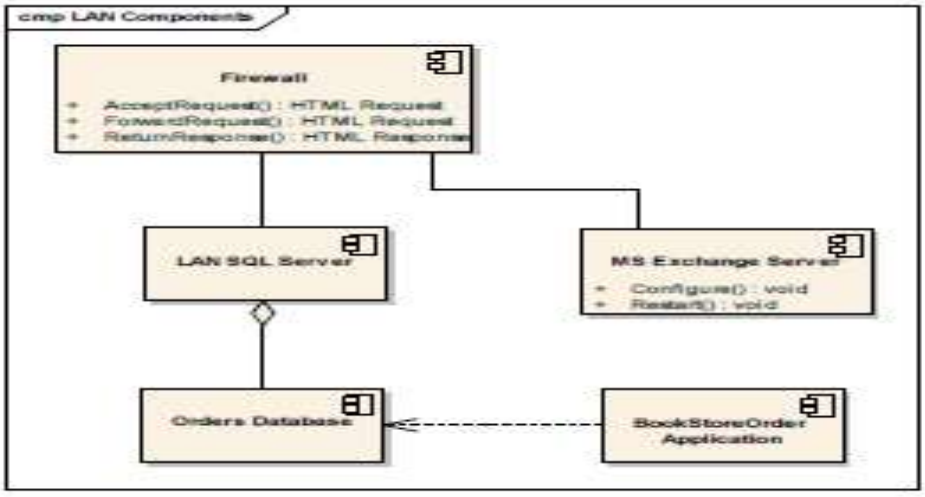
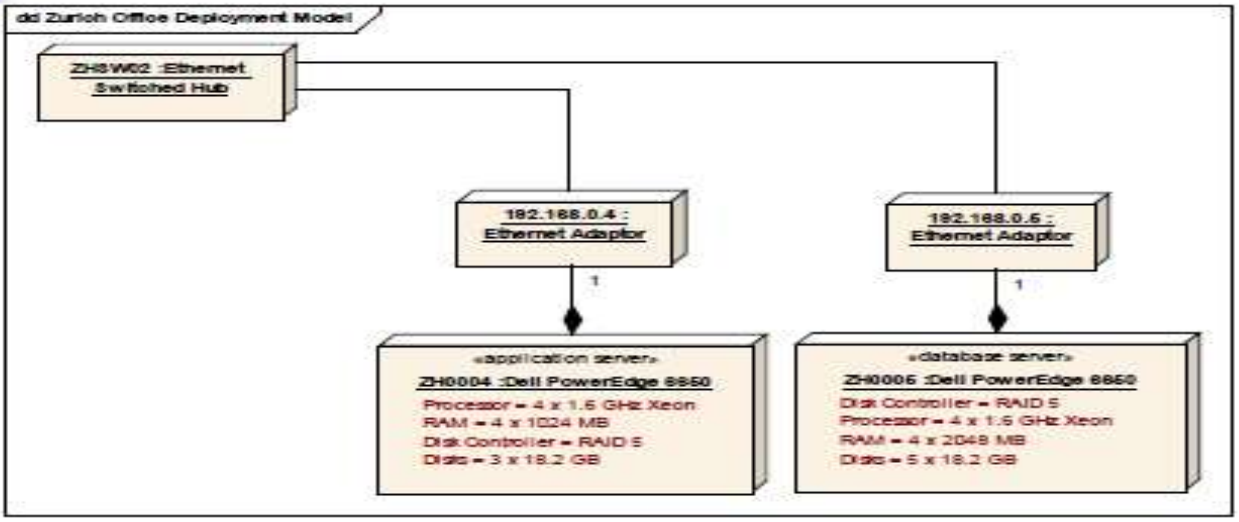


Diagram nasazení



Sekvenční diagram

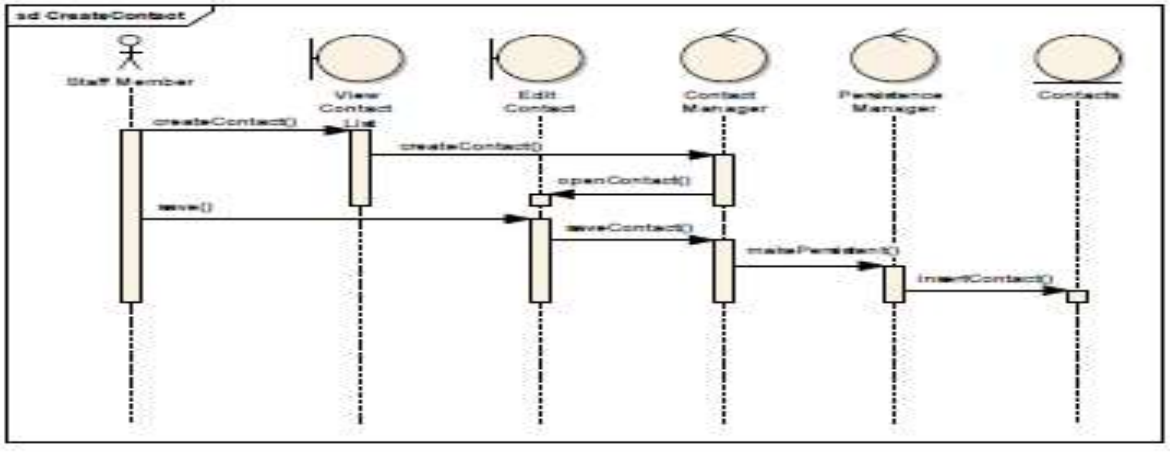
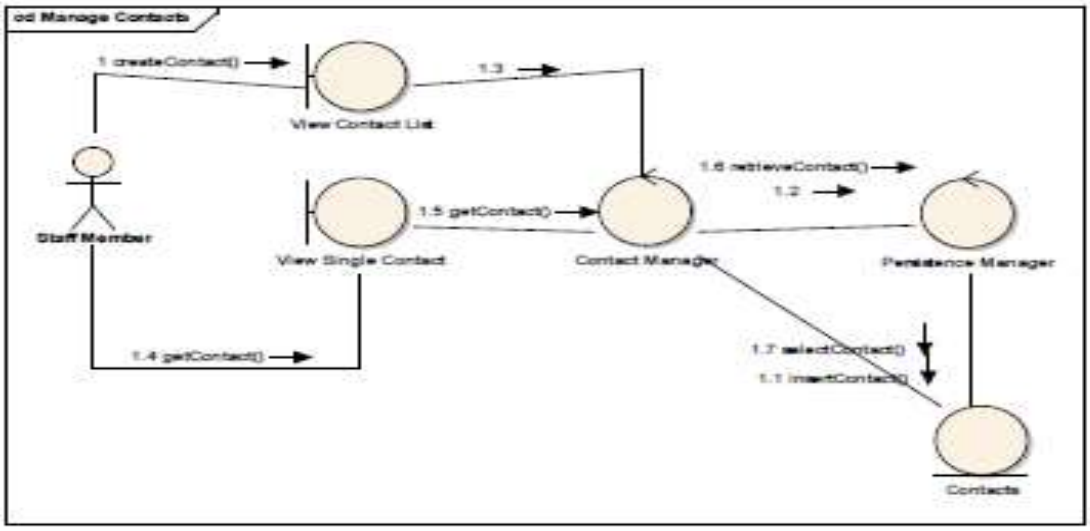


Diagram komunikace



Stavový diagram

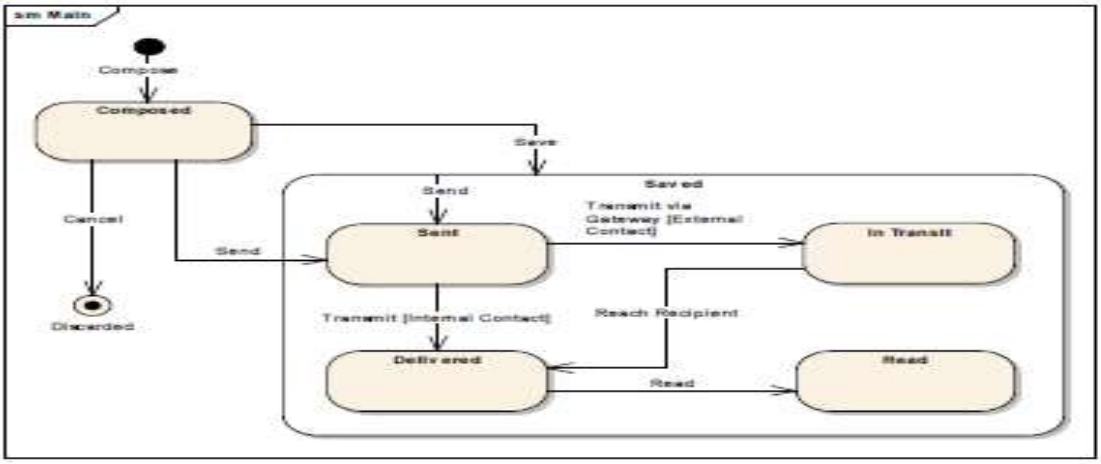
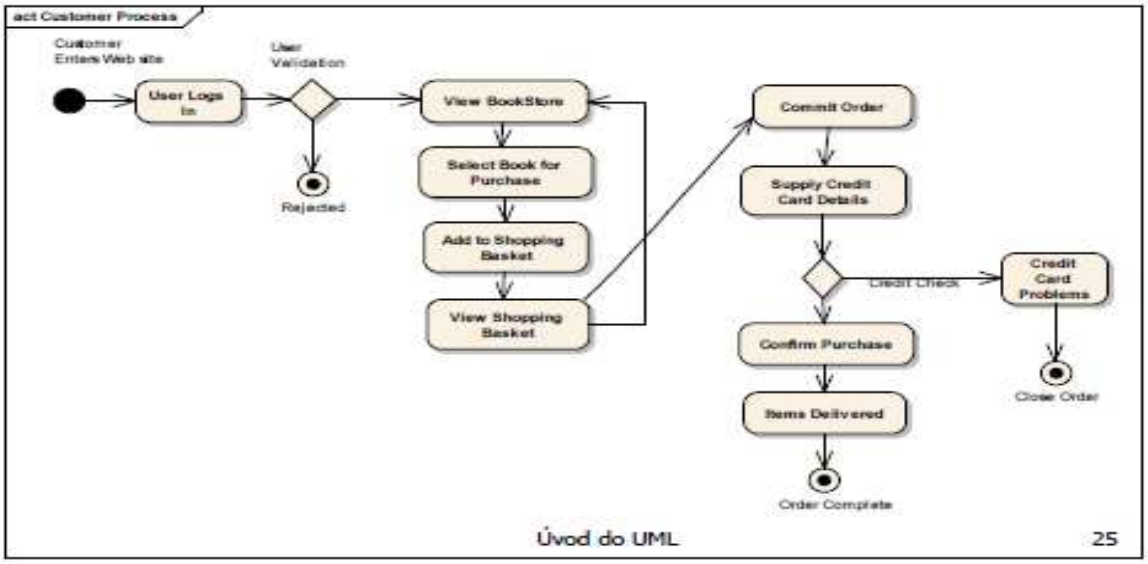


Diagram aktivit

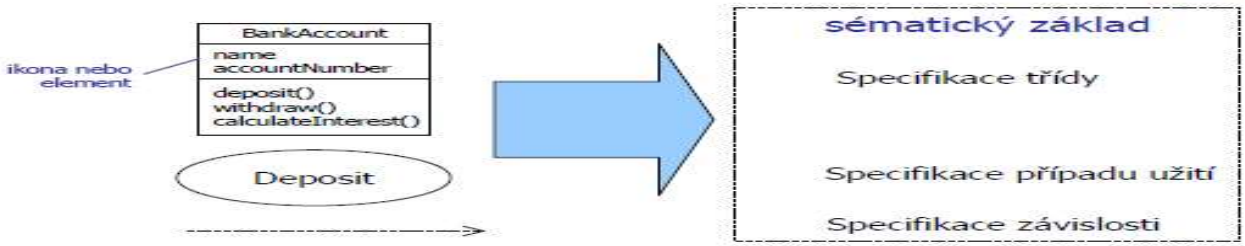


Společné mechanizmy

- **Specifikace**
- **Ornamenty (adorments)**
  - **Podskupiny**
- **Mechanizmy rozšiřitelnosti**

Specifikace

Specifikace



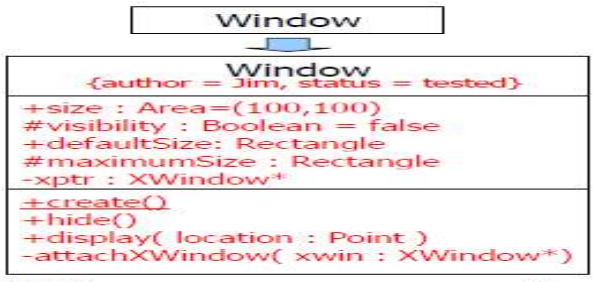
- Pro každý element existuje textová specifikace popisující jeho sémantiku a syntaxi
- Specifikace tvoří sémantický základ modelu

Úvod do UML 27

Ornamenty (Adornments)

Ornamenty (Adornments)

- Každý element modelu má základní podobu, ke které je možné přiřadit upřeňující *ornamenty*
- Ornamenty zobrazovány jen pro upřesnění či zvýraznění vlastnosti elementu

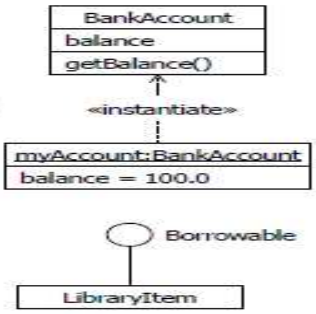


Úvod do UML 28

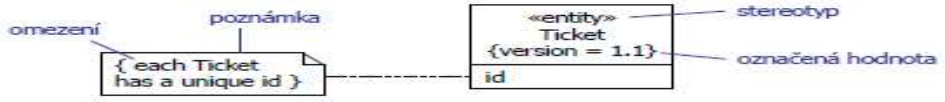
Podskupiny

Podskupiny

- Klasifikátor a instance
  - Klasifikátor je abstrakce, instance je konkrétní projev této abstrakce
  - Nejběžnější forma třída/objekt
  - Obecně instance mají stejnou notaci jako třídy, ale jméno podtrženo have the same notation as classes, but the instance name is underlined
- Rozhraní a implementace
  - Rozhraní definuje kontrakt a implementace představuje konkrétní realizaci tohoto kontraktu



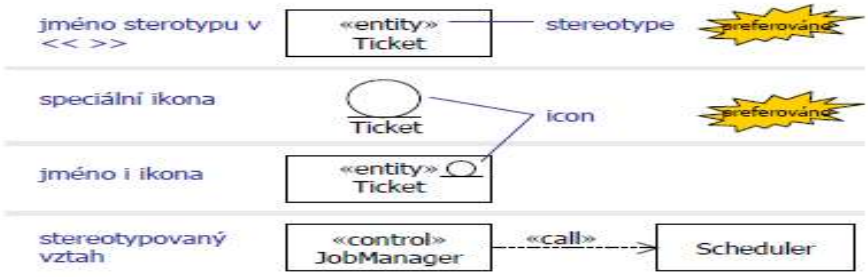
Mechanizmy rozšiřitelnosti



- Stereotypy
  - umožňuje definovat nový UML element na základě existujícího
  - Sémantiku si definujeme sami
  - Zápis: «stereotypeName»
- Omezení (Constraints)
  - Rozšiřují sémantiku elementu o nová pravidla
  - Zápis: { pravidlo }
- Označené hodnoty (Tagged values)
  - Umožňují přidat novou ad-hoc informaci elementu
  - Zápis: { tag1 = value1, tag2 = value2 ... }

připojeny ke stereotypu

Možné syntaxe stereotypu



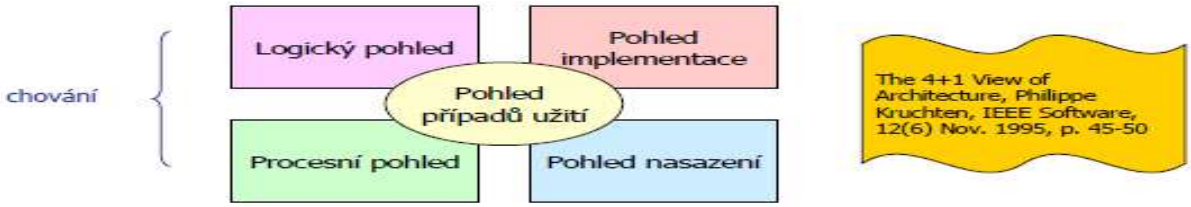
- Stereotyp definuje nový element – musíme popsat jeho sémantiku
- Každý element modelu může mít více sterotypů

Profily UML

- **UML pro specifické účely**
- **UML prfil je kolekce stereotypů, omezeních a označeních dat**
  - **Označené hodnoty a omezení jsou asociovány se stereotypy**

Architektura

Organizační struktury softwarového systému)



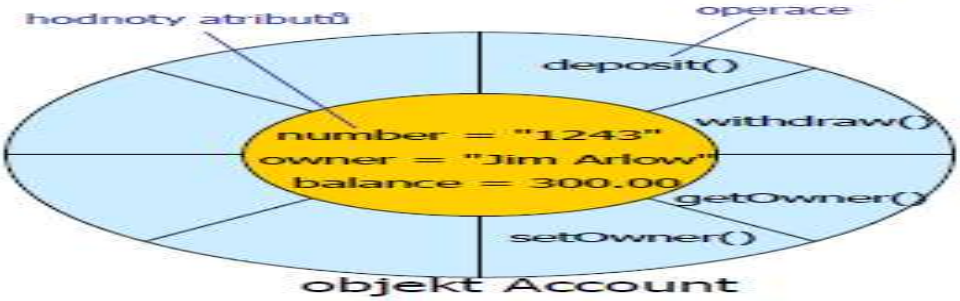
Co jsou to objekty

- Objekt – znovupoužitelná jednotka skládající se z dat a funkcí zabalených dohromady – Objekt zapouzdřuje data
- Každý objekt – instance nějaké třídy
- Třídy definuje společné rysy (vlastnosti a operace) společné pro všechny instance

- Objekt má:
  - **Atributy** – datová část
  - **Operace** – chování
- Všechny objekty mají:
  - **Identitu** – objekt je jednoznačně identifikovatelný
  - **Stav** – určený aktuálními hodnotami atributů a vazbami na jiné objekty
  - **Chování** - množina operací, které může daný objekt aktuálně provádět

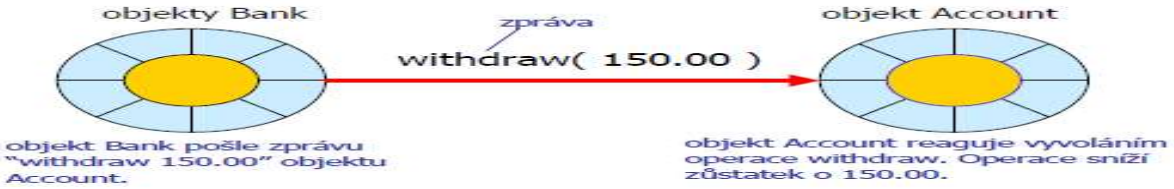
Zapouzdření

- Data schovaní uvnitř objektu
- Přístup pouze pomocí operací

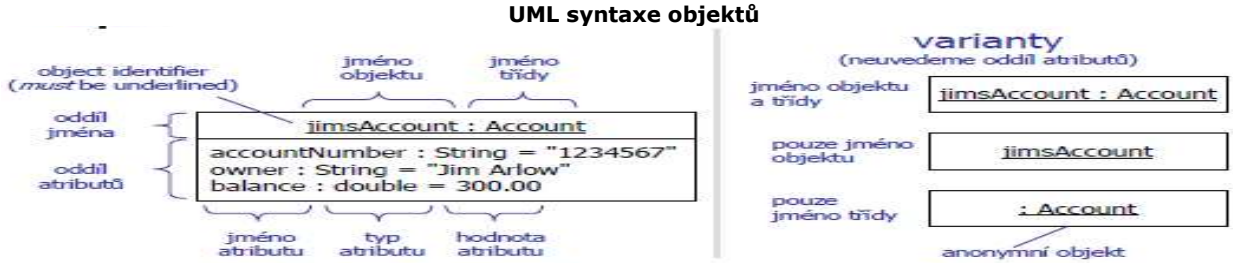


Posílání zpráv

- V OO systémech posílají si objekty zprávy prostřednictvím linků
- Poslání zprávy způsobí vyvolání metody cílového objektu



UML syntaxe objektů



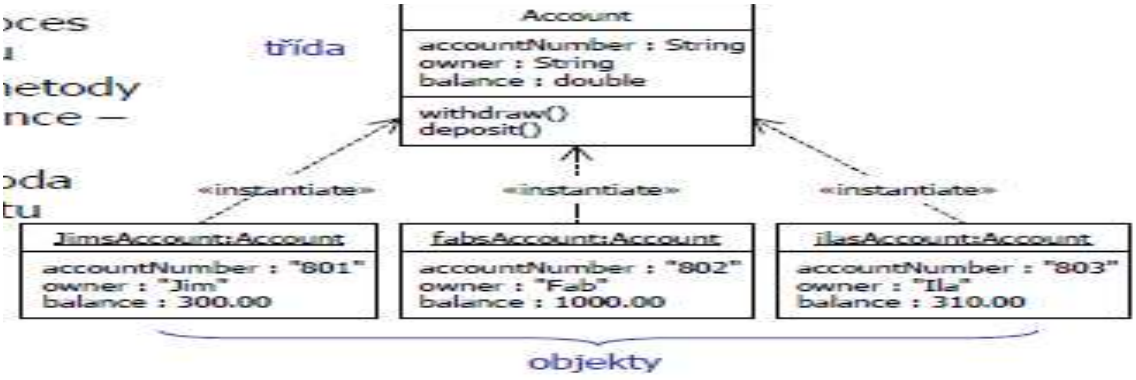
- Všechny objekty dané třídy mají stejnou množinu operací – nezobrazují se u objektů ale u tříd).
- Typy atributů pro přehlednost obvykle vynechány
- Konvence pojmenování:
  - objekty a atributy lowerCamelCase
  - jméno třídy UpperCamelCase

Co jsou to třídy

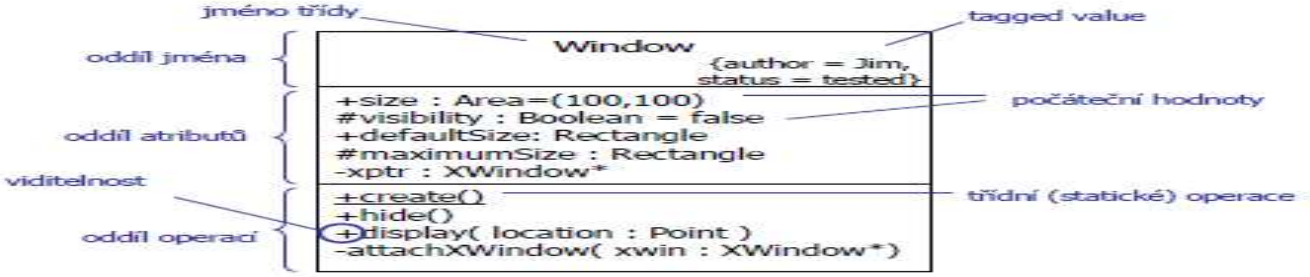
- Každý objekt je instancí třídy
- Třída definuje typ objektu
- Třídy umožňují modelovat množinu objektů se stejnými vlastnostmi – třída představuje šablonu na objekty:¨
  - Stejné atributy
  - Stejnou množinu operací
  - Obecné různé hodnoty atributů
- Klasifikace (rozdělení na třídy) umožnuje udržet si organizovaný pohled na svět
- Představujte se třídy jako:
  - Razítka
  - Formičky na cukroví

Třídy a objekty

Objekty – instance tříd
Instance – proces vytvoření objektu
Třídy poskytují metody na vytvoření intance – konstrukce
Konstruktor metoda třídy nikoli objektu



UML natoce třídy



- Třídy pojmenované UpperCamelCase
- Pojmenovávejte jména podstatných jmen
- Nepoužívajte zkratky

Oddíl atributů

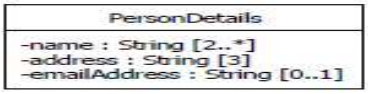
Visibility name : type multiplicity = intialValue

Povinné

- Vše kromě jména je volitelné
- initialValue je hodnota, kterou atribut dostane při instanci
  - Atributy pojmenované atributy podstatnými jmény
    - Nepoužívejte zkratky
    - Atribut může mít
      - Stereotyp jako prefix
      - Seznam tagged values jeho postfix

Viditelnost

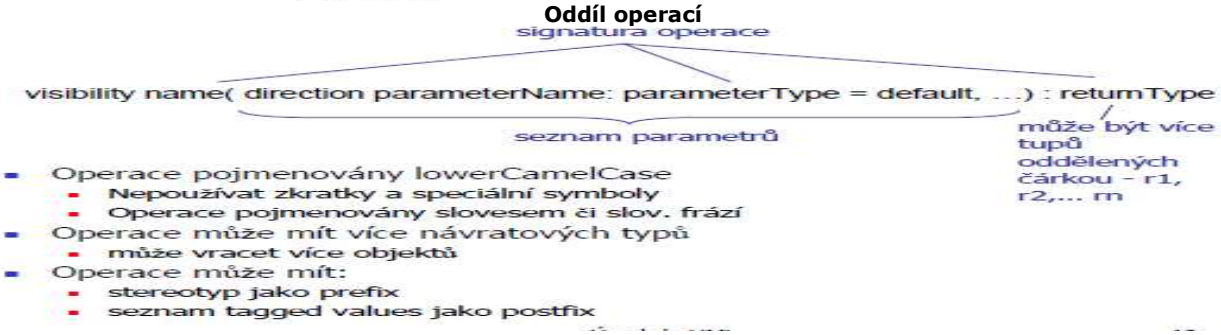
Symbol	Jméno	Kdo má přístup
+	public	Každý
-	private	Pouze operace uvnitř třídy
#	protected	Pouze operace uvnitř třídy nebo potomci třídy
~	package	Každý ze stejného balíčku či podbalíčku uvnitř třídy



Násobnost (multiplicita)

- Umožnuje modelovat kolekce věcí

Oddíl operací



Druhy parametrů

druh parametru	význam
in	vstup do operace operace neměni jeho hodnotu
out	je do něj uložen výstup operace
inout	slouží jako vstup operace ho může změnit
return	návratová hodnota operace

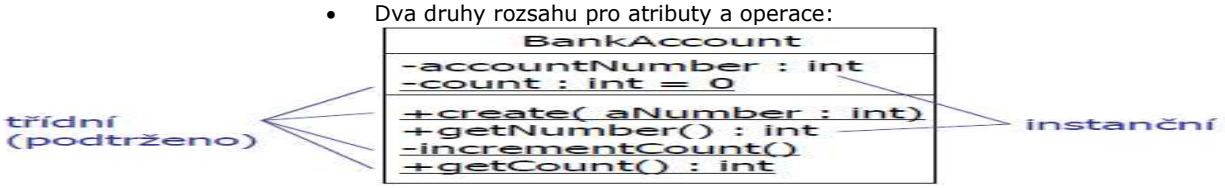
example of multiple return values:

```
maxMin( in a: int, in b:int return maxValue:int return minValue:int )
maxMin( in a: int, in b:int ):int,int

max, min = maxMin( 5, 10 )
```

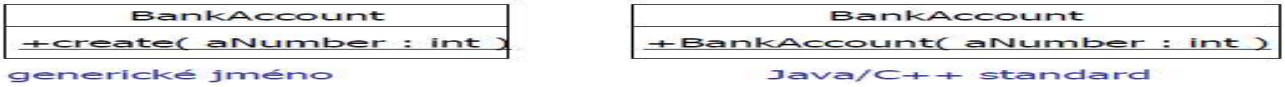

Rozsah platnosti

- Dva druhy rozsahu pro atributy a operace:



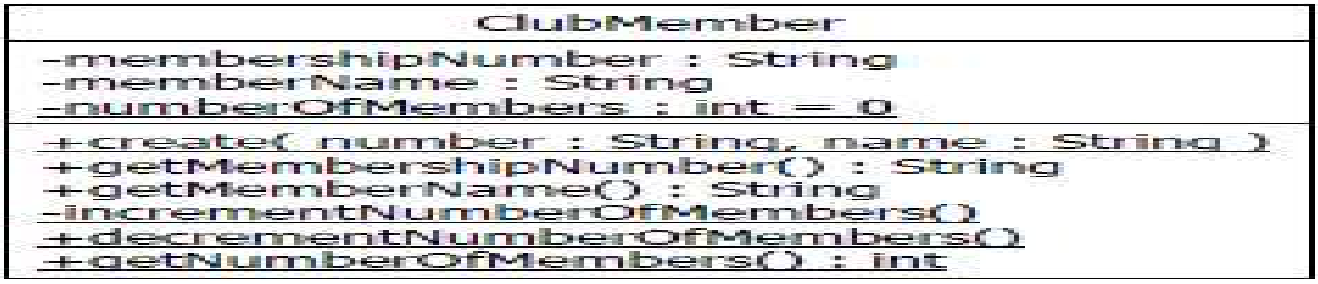
Konstrukce objektu

- Třída poskytuje (statické operace) jeden či více konstruktorů



Příklad ClubMemeber

- Každý objekt ClubMeter má svoji kopii atributu membershipNumber
- Atribut numberOfMembers existuje jednou a je sdílen všemi instancemi třídy ClubMember
- Počítání instancí



Co

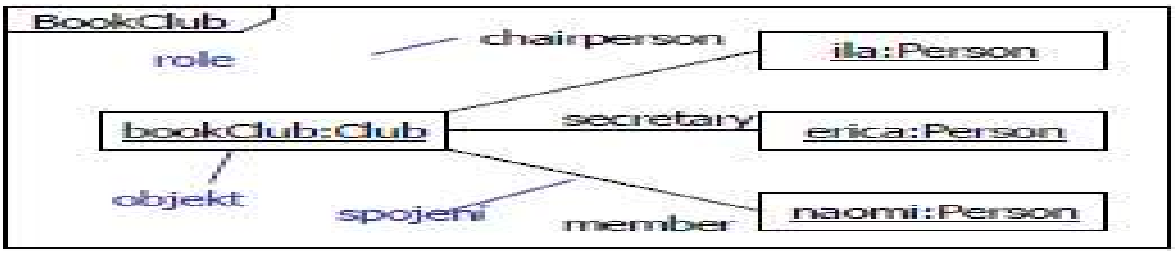
Co je to vztah

- Spojení mezi modelovanými elementy
  - Zde:
    - Spojení mezi instancemi
    - Asociace mezi třídami
      - Agregace
      - Kompozice
      - Asociační třídy

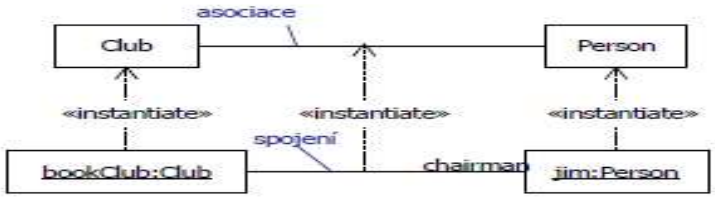
Co je to spojení?

- Spojení sémantická vazba mezi instancemi
  - Pomocí spojení objekty komunikují
    - Zprávy posílání pomocí spojení
    - Zprávy vyvolávají operace
- OO prog. Jazyky implementují spojení jako reference či pointery.

Spojení objektů



Co je to asociace



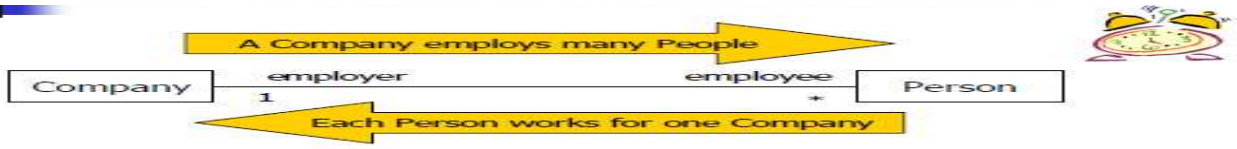
- asociace – vztahy mezi třídami
- potenciální existence spojení mezi instancemi

Syntaxe asociace

- pojmenování asociace:
  - jméno
  - role
- asociaci čteme: "Company employs many Person(s)"

Úvod do UML 55

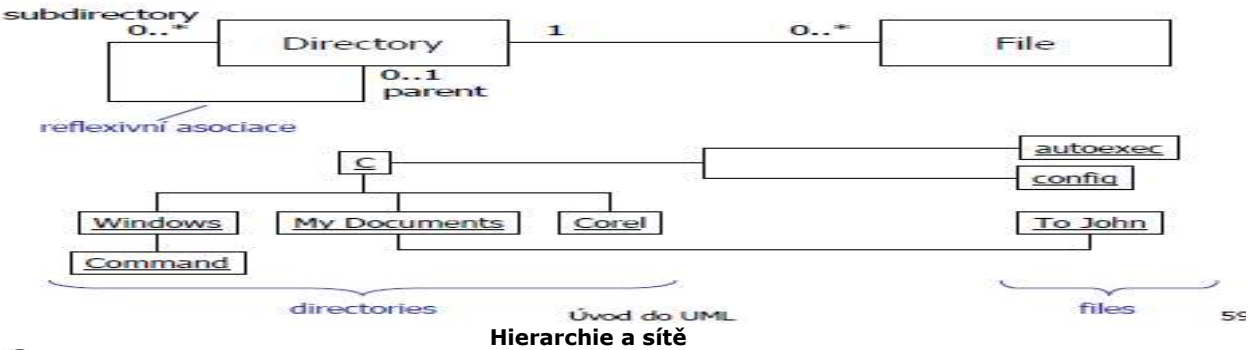
Multiplicita



- specifikuje počet objektů participujících na vztahu
- není povinná

multiplicity syntax: minimum..maximum	
0..1	nula nebo 1
1	právě 1
0..*	nula či více
*	nula či více
1..*	1 či více
1..6	1 až 6

Reflexivní asociace

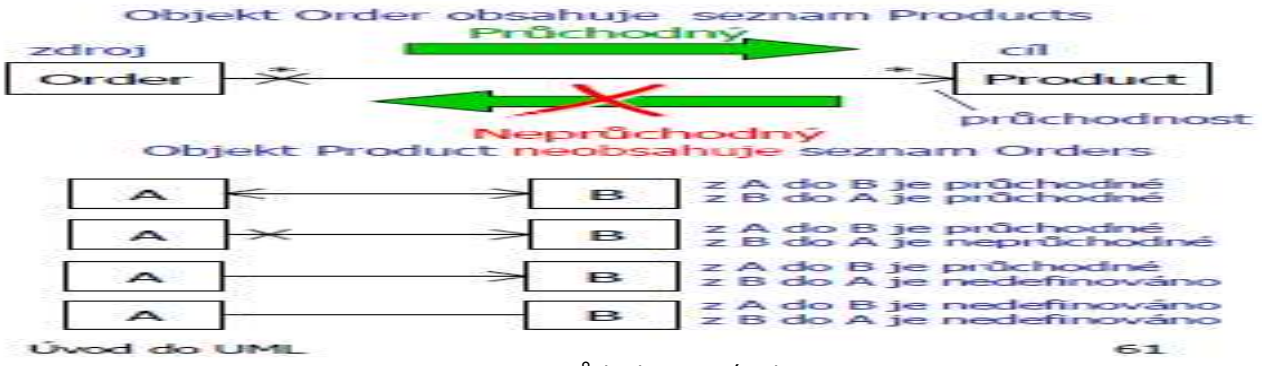


Úvod do UML 59

Hierarchie a sítě

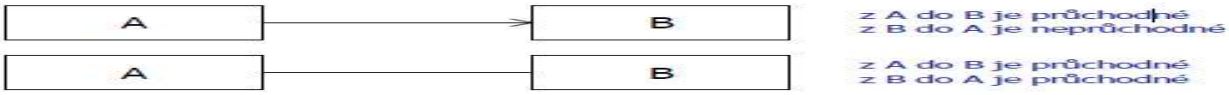
Průchodnost

- Průchodnost vyjadřuje, jakým směrem lze posílat zprávy
  - Jaký objekt má odkaz na který



Průchodnost – zásady

- 3 možné způsoby:
  - Znázorňuje průchodnost explicitně všude
  - Neznázorňuje průchodnost nikde
  - Neznázorňuj křížky
- Obousměrné nemají šipky - standart
- Jednosměrné mají šipku - standart



Asociace vs Atributy

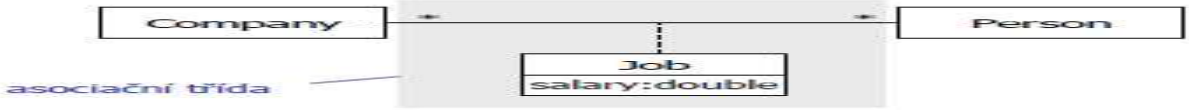
- Při jednosměrné asociaci s rolí může být nahrazena atributem
  - Vyjadřují to samé
  - Kdy použít asociaci:
    - Cílová třída je důležitá pro model
    - Cílová třída je nestandardní
  - Kdy použít atribut:
    - Cílové třída je standardní nebo primitivní typ

Asociace třídy

Pokud je osoba zaměstnána společností, kam umístíme plat?

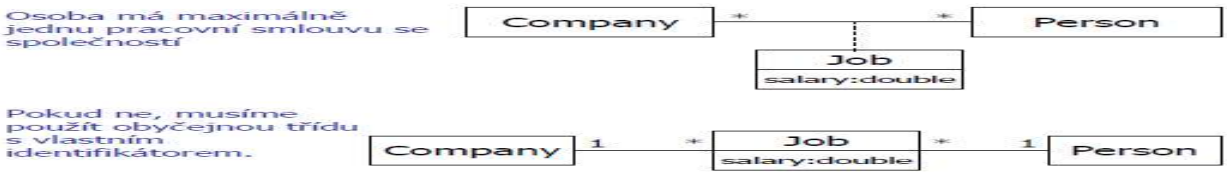
- Do person – různé platy v různých zaměstnáních
  - Ne do company – různí lidé různé platy
    - Plat je vlastnost vlastního vztahu

Syntaxe asociační třídy

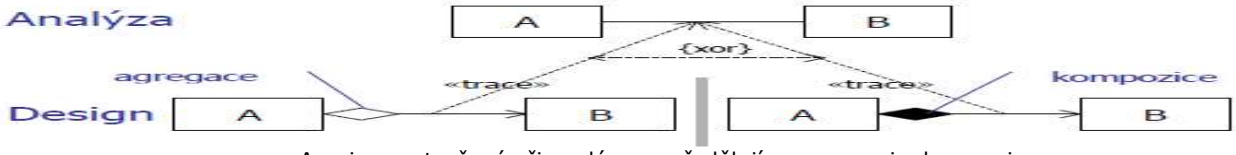


- Jedna aociační třída pro jedno spojení
  - Má
    - Atributy
    - Operace
  - Jako obyčejná třída
  - Identifikovaná párem asociujících objektů

Použití

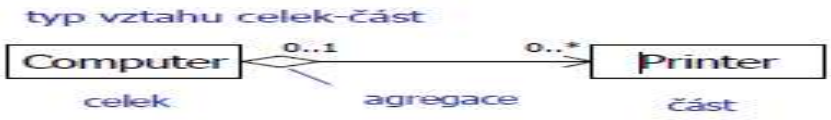


Agregace a kompozice

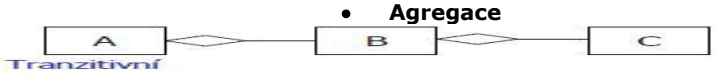


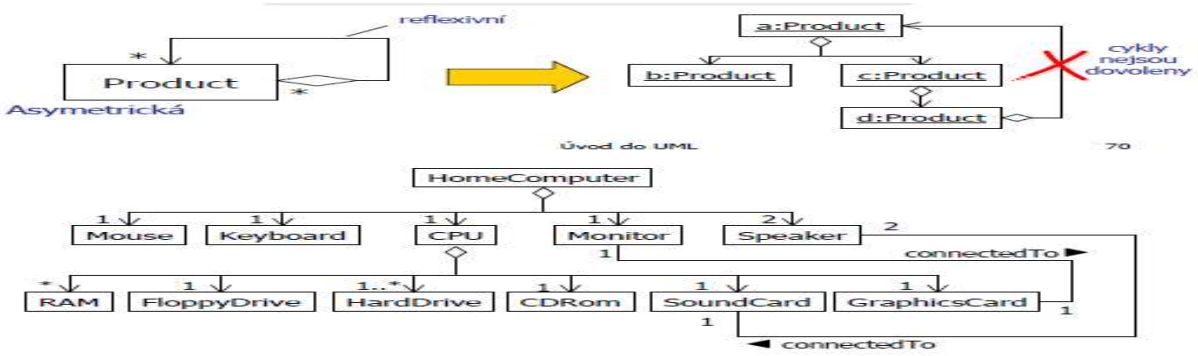
- Asociace vytvořené při analýze se předělají na agregaci a kompozice
- Třeba doplnit průchodnost, násobnost a jména rolí.

Agregace



- Celek a část nejsou na sobě existenčně závislé
- Části mohou být sdíleny více celky
- **Agregace**





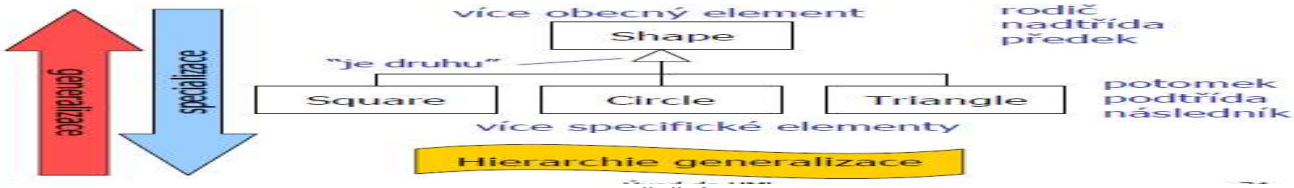
Kompozice

- Části nejsou sdílené
- Celek je zodpovědný za instrukci a destrukci částí
- V momentě zániku celku zanikají i jeho části
- Kompozice tranzitivní a asymetrická

Generalizace

- Vztah mezi obecným a více specifickým elementem§
- Více specifické elementy konzistentní s obecným, ale obsahuje více informací§
- Instance více specifického elementu použitelná všude, kde je použitelná obecná

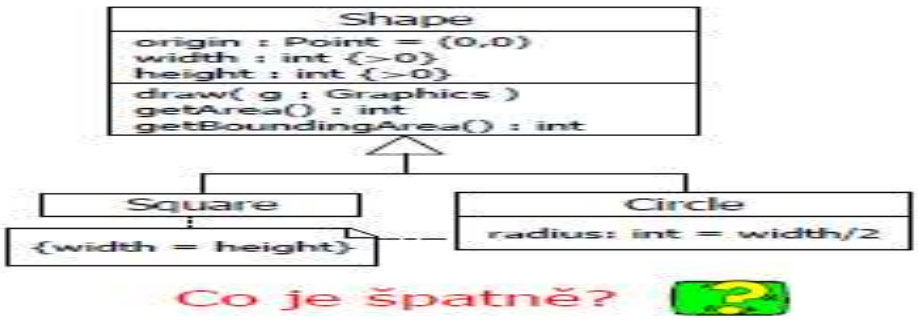
Generalizace tříd



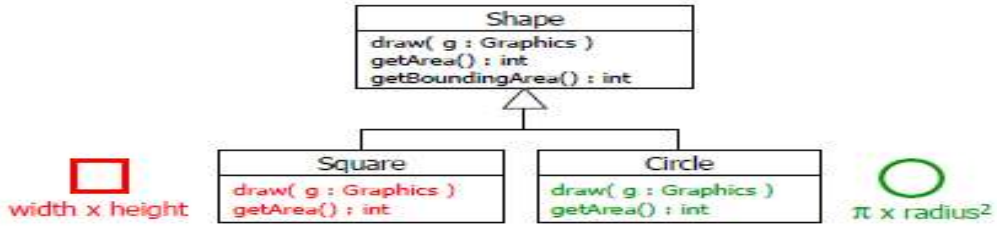
Dědění

Dědění

- Podtřídy dědí všechny vlastnosti nadtřídy:
  - Atributy
  - Operace
  - Vztah
  - Stereotypy, tagy, omezení
- Podtřída přidává nové prvky
- Podtřída může změnit implementaci operací

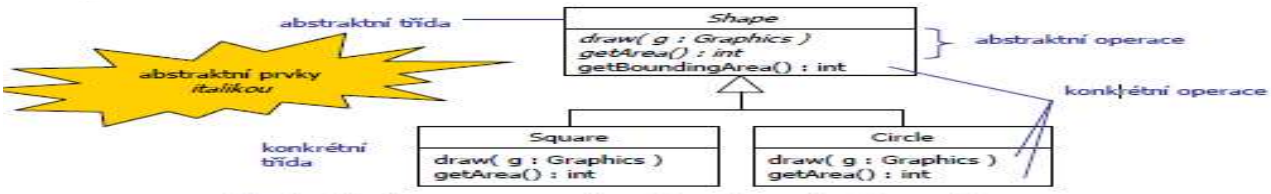


Překrývání



- Podtřída často potřebuje změnit chování definované v předkovi
- Při přetížení signatura v potomkovi musí být identická s předkovou
  - Jména parametrů nehrají roli

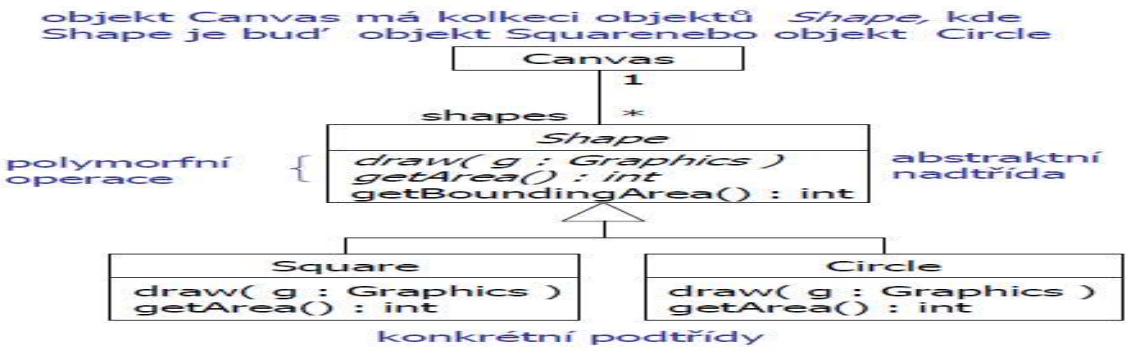
Abstrakce operace



- Abstrakce operace – signatura bez implementace
- Třída abstraktní pokud má nějaké abstraktní operace
- Od abstraktní třídy není možné vytvořit instance
- Potomek abstraktní třídy také abstraktní pokud neimplementuje abstraktní operace.

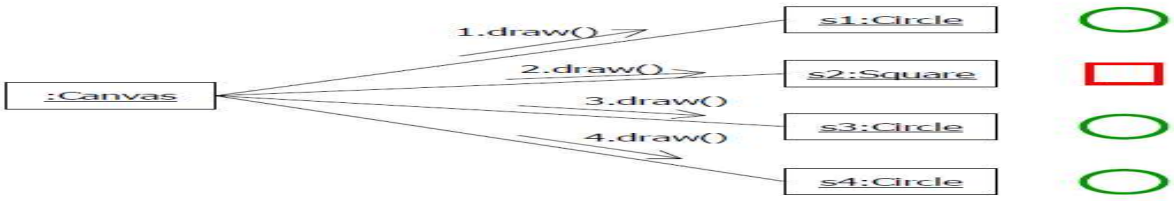
Polymorfismus

- Polymorfismus = „více forem"
  - Polymorfní operac – více implementací
    - Shape::draw() and
      Shape::getArea ()



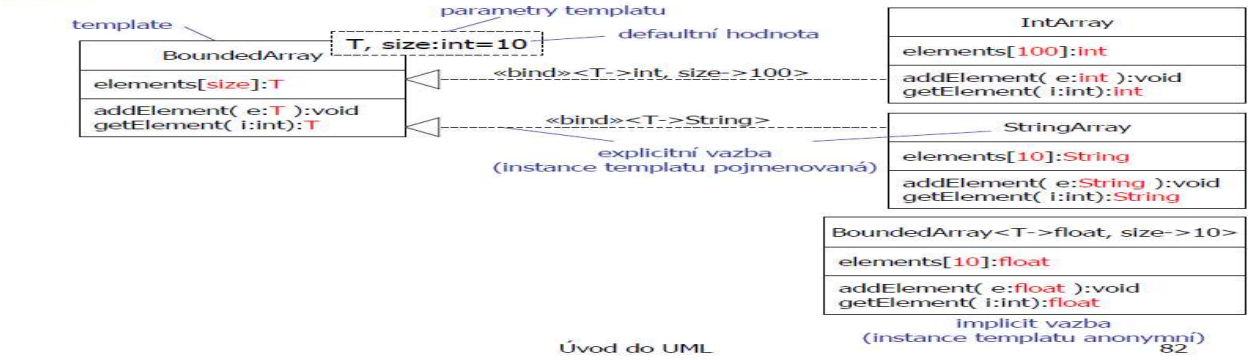
Co se stane?

- Každá třída má svou implementace operace draw ()
- Každý objekt realizuje operaci draw() po svém



Příklad – BankAccount

Syntaxe generických tříd



Vnoření třídy

Vnořená třída je třída definovaná uvnitř jiné třídy

Rozhraní (Interface)

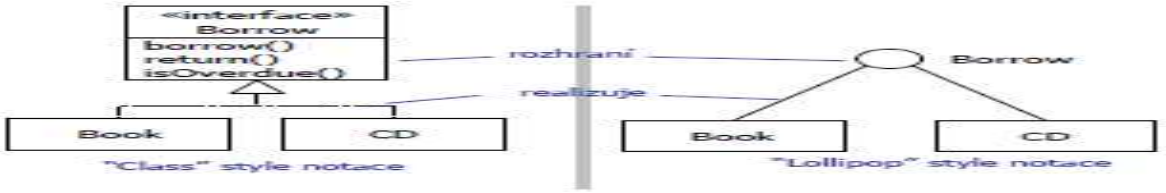
- Speciální druh (abstraktní) třídy
- Specifikuje pojmenovanou množinu vlastností
- Účel – oddělení specifikace funkčnosti od implementace
- Rozhraní definuje kontrakt, který musí implementující klasifikátory realizovat

Kontrakt

Rozhraní specifikuje	Realizující klasifikátor
Operace	Musí mít operaci se stejnou signaturou a sémantikou
Atribut	Musí mít veřejné operace pro čtení/modifikace atributu.
Asociace	Implementující klasifikátory musí mít mezi sebou stejnou asociaci jako rozhraní.
Omezení	Must support the constraint
Stereotypy	Má stereotyp
Tagged value	Má tagged value
protokol	Realizuje protokol

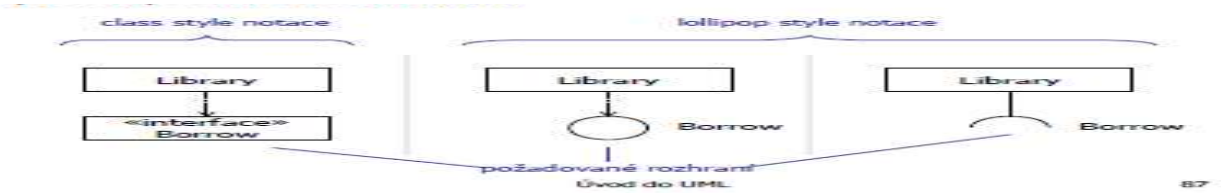
Syntaxe poskytovaného rozhraní

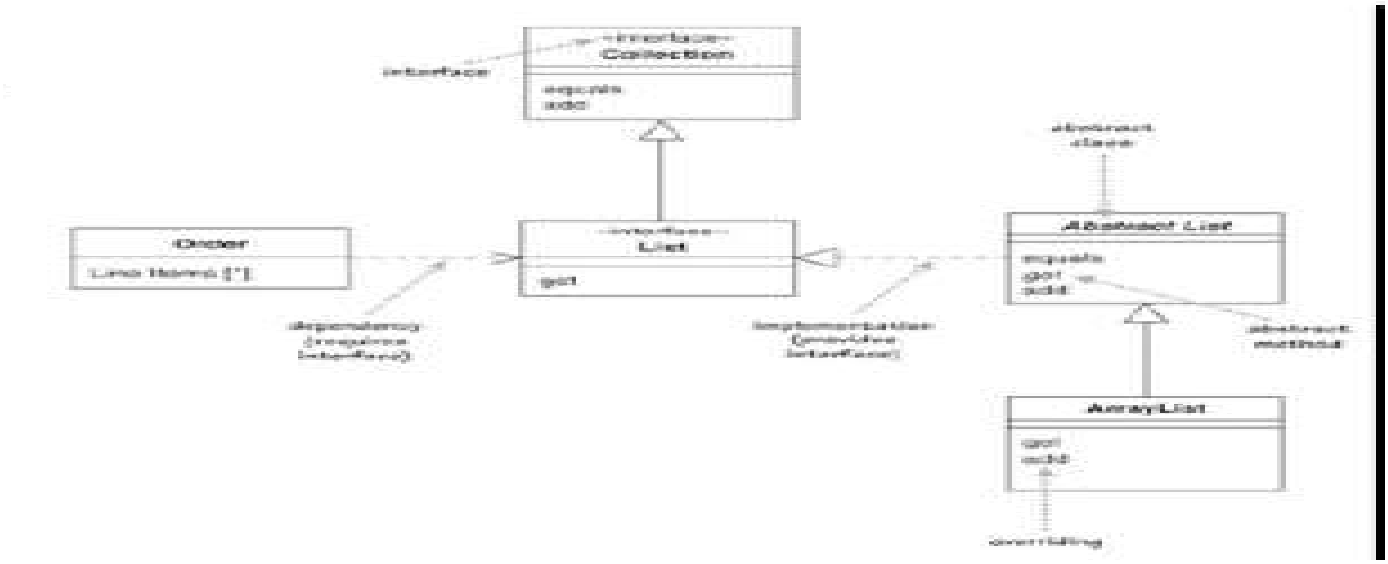
- Poskytování rozhraní – klasifikátor ho implementuje



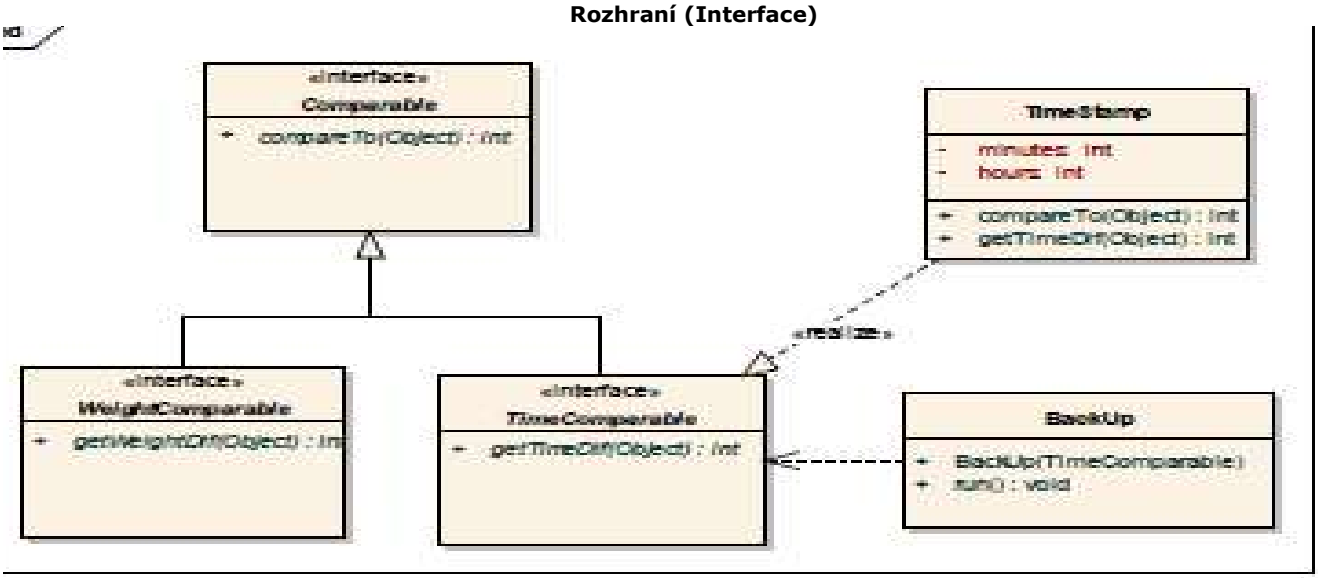
Syntaxe požadovaného rozhraní

- Požadované rozhraní - Klasifikátor používá metody předepsané rozhraní



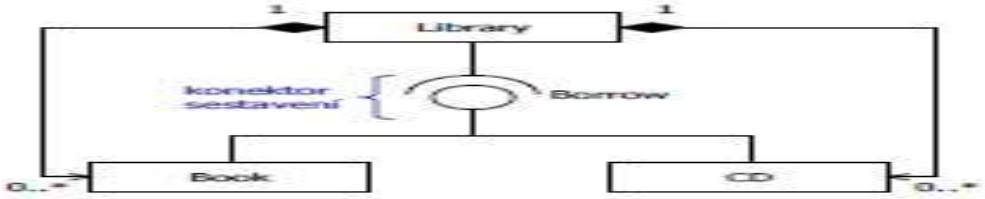


Rozhraní (Interface)



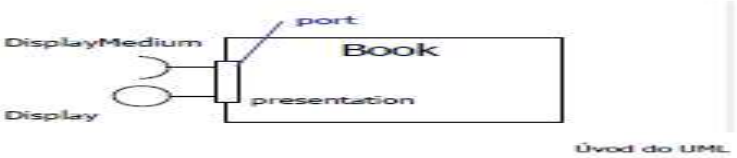
Konektor sestavení

- Poskytované a požadované rozhraní možné spojit konektor



Porty – organizace rozhraní

- Bod interakce mezi klasifikátorem a rozhraním
- Seskupuje logicky související rozhraní
- Port má typ daný poskytovaným a požadovaným rozhraním
- Může mít jméno



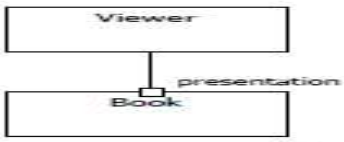


Diagram tříd

Diagram tříd

- Vše předchozí vyjma instancí jsou stavební prvky diagramu tříd
- Diagram tříd nezobrazuje objekty

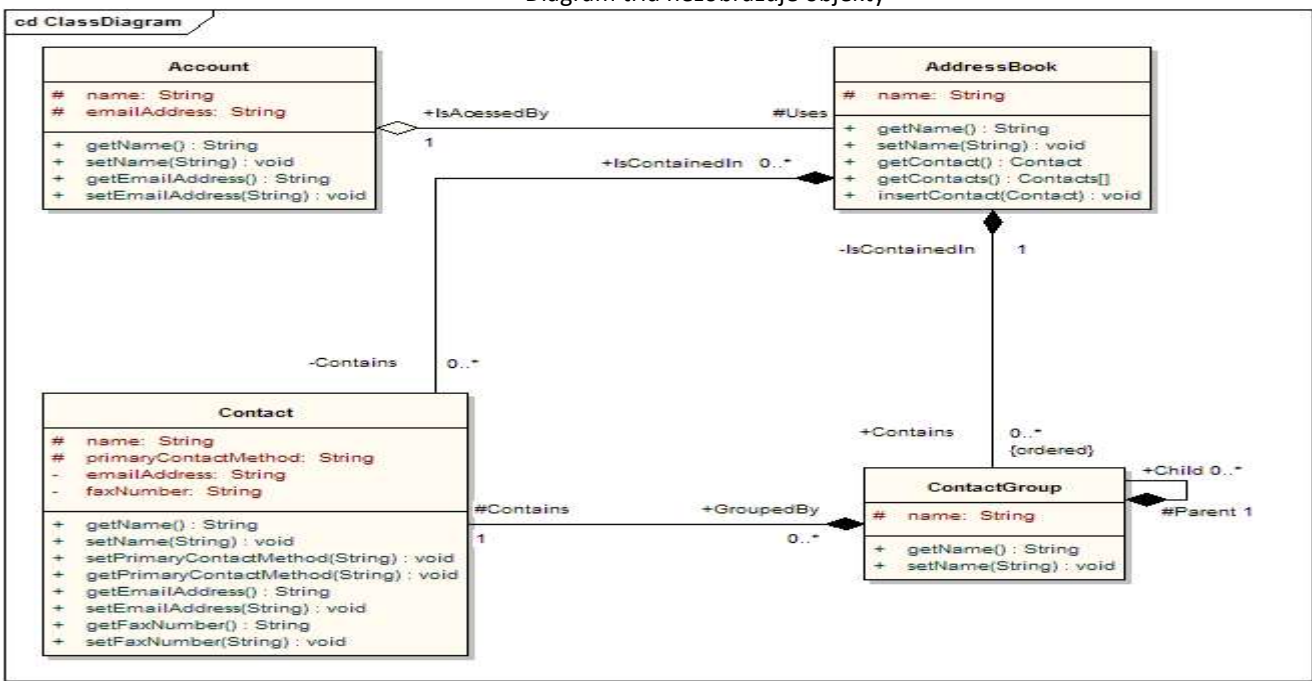
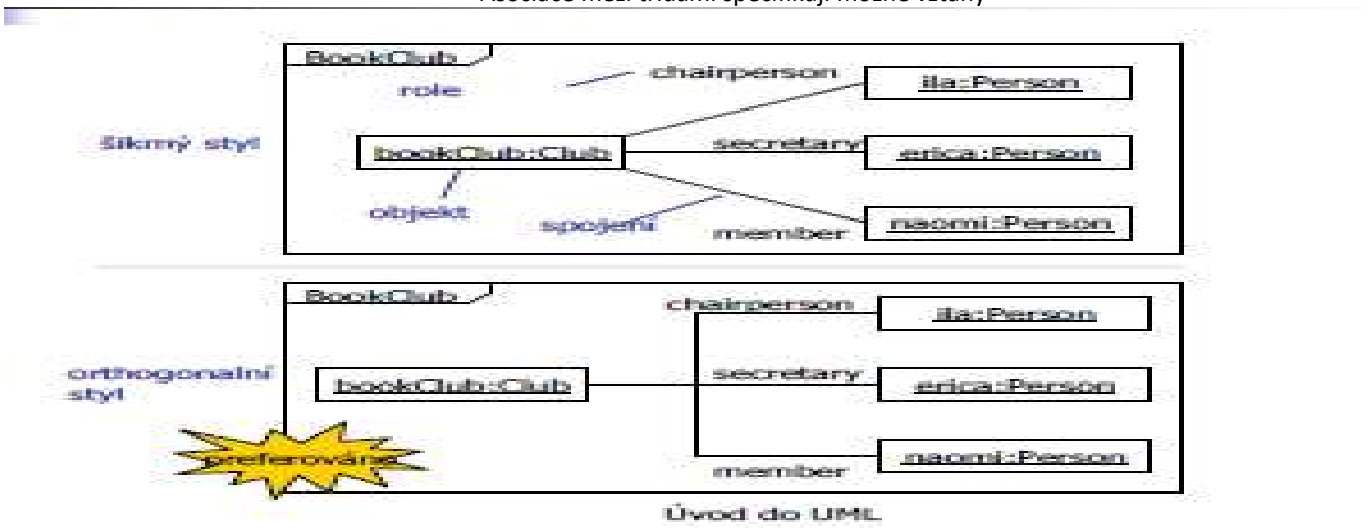
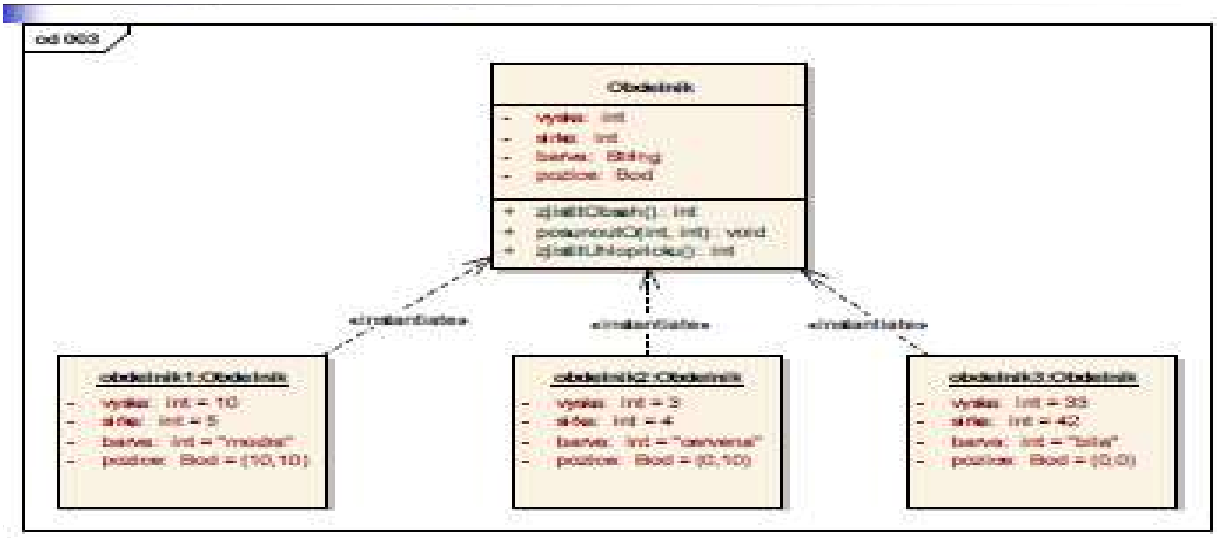


Diagram objektů

Diagram objektů

- Zachycuje instanci tříd a jejich momentální vztahy v určitém momentu
- Může obsahovat i třídy – vztah instance/třídy
- Umožnuje vyjádřit role zobrazovaných objektů
- Asociace mezi třídami specifikují možné vztahy





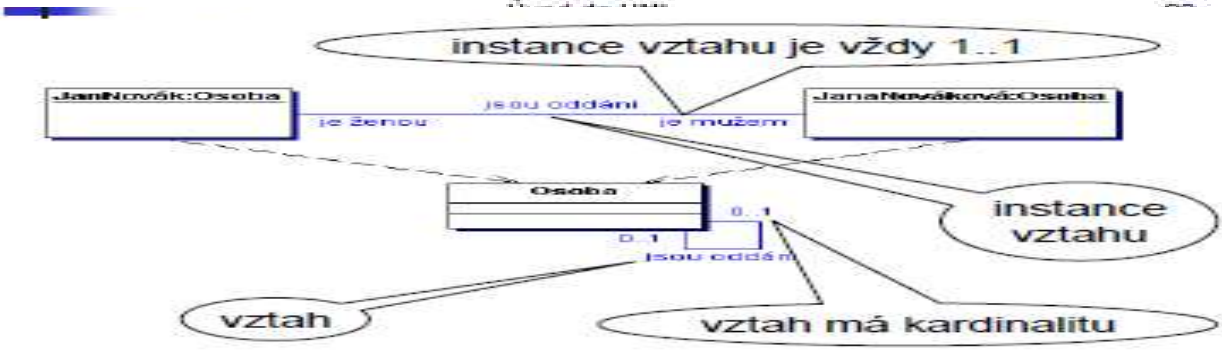
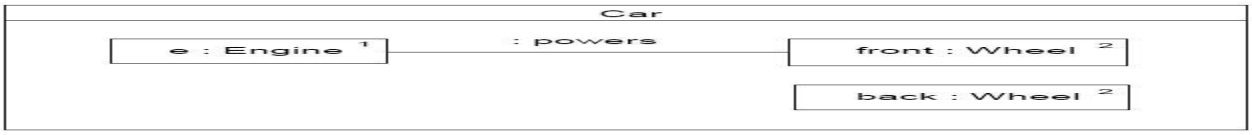


Diagram struktury

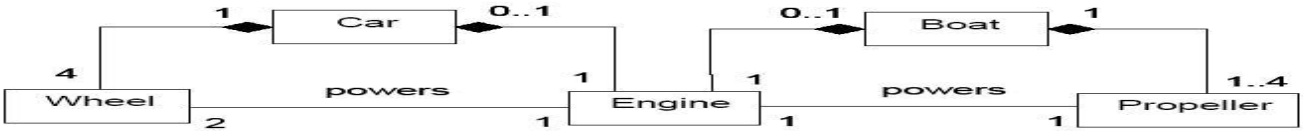
Diagram struktury

- Ukazuje interní strukturu klasifikátoru včetně jeho bodů interakce s okolím
- Ukazuje konfiguraci a vztahy částí klasifikátoru zajišťující jeho chování
- Klasifikátory především
  - Třídy
  - Komponenty



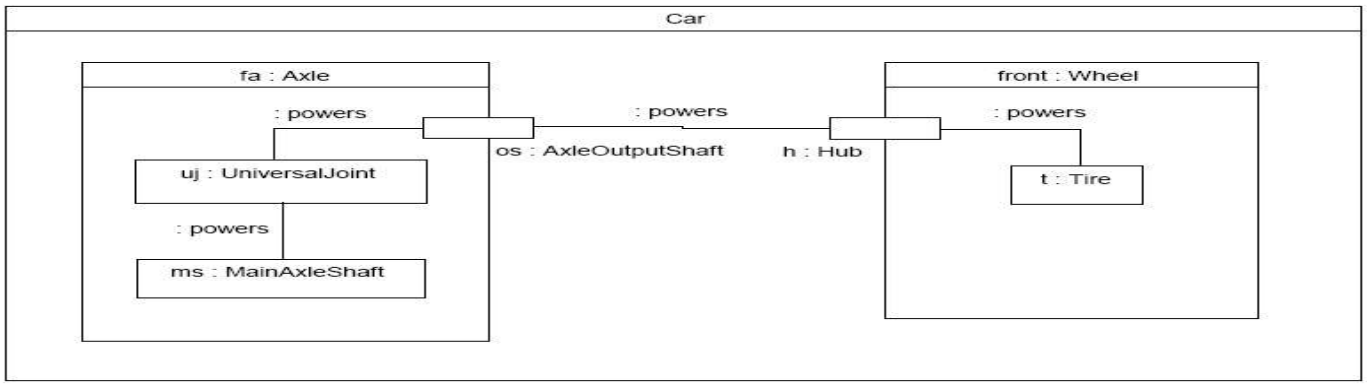
- V každém autě je jeden motor, dvě přední a dvě zadní kola
- Motor pohání přední kola
- Motor nepohání nic jiného

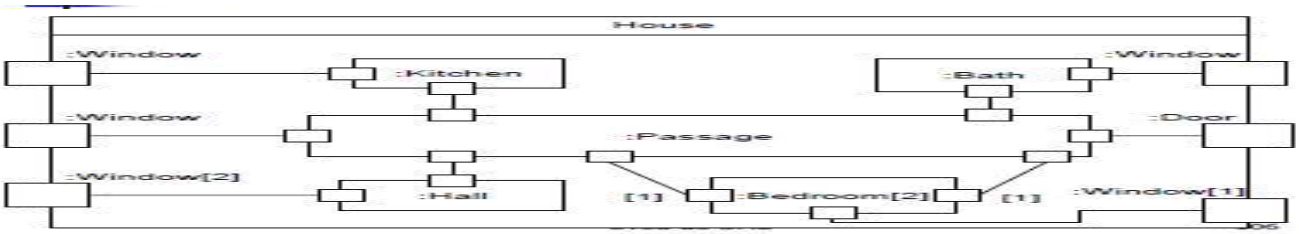
- V každém člunu je jeden motor a několik lodních šroubů
- Motor pohání lodní šrouby
- Motor nepohání nic jiného
- Totéž pomocí tříd?



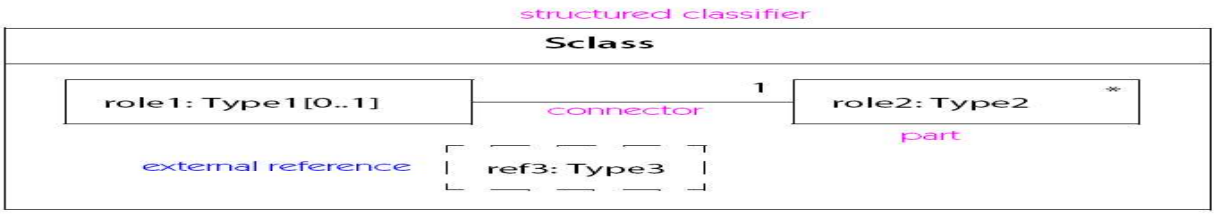
- Motor v jednom autě může pohánět kola v jiném autě
- Každý motor pohání kola i lodní šrouby

- Motor může pohánět libovolná dvě kola



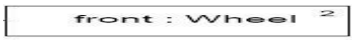


Notace



Část (part)

- Instance nějakého klasifikátoru
  - Představuje roli
    - Vlastnosti
      - Jméno role
      - Typ (klasifikátor)
      - Multiplicitu
  - Např. front:wheel[2]



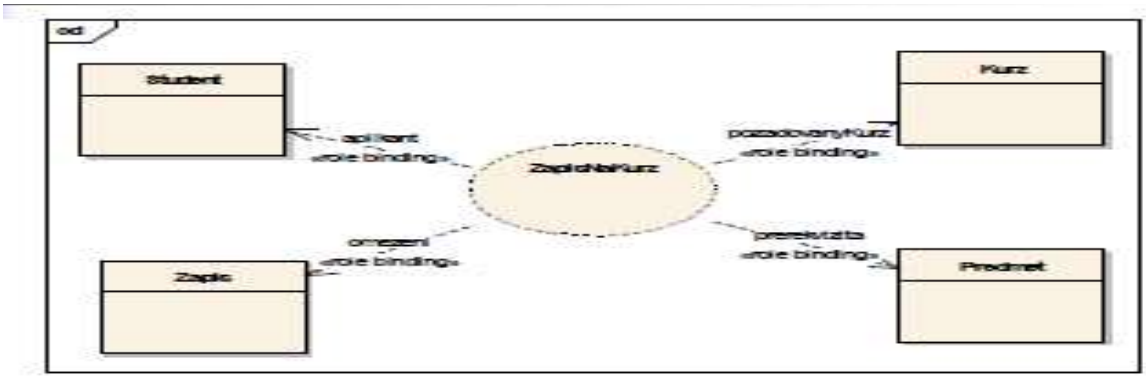
Konektor (Connector)

- Specifikuje instanci vztahu uvnitř klasifikátoru
  - Vztah mezi instancemi v rolích
    - Rozdíl
      - Asociace – vztah mezi klasifikátory
      - Konektory – vztah mezi rolemi
  - Např. motor pohání přední kola (:powers)

Port

- Bod interakce mezi klasifikátorem a okolím
- Zprávy jsou směrování na instance portů, na vlastní objekty
  - Delegace
    - Rozpoznání zprávy
    - Přesměrování na zodpovědnou část
  - Porty vznikají a zanikají spolu s objekty

Kolaborace



- Cíle
  - Kdo se na spolupráci podílí
  - Jakou při tom hraje roli
  - Volitelně: jaké atributy jsou užity při spolupráci
  - Nejde o kompletní definici třídy

Diagram komponent

Komponenty

- Modulární část systému zapouzdřující svůj obsah a jejíž projev je nahraditelný v daném prostředí
  - „černá skřínka“ projevující se pomocí požadovaných a poskytovaných rozhraní
    - Zastupuje cokoli, z čeho lze vytvořit objekt za běhu systému

Rozhraní and CBD

- Rozhraní jsou klíčová pro component based development (CDB)
  - CBD je konstrukce SW výměnných plug-in částí:
    - Plug – poskytované rozhraní
    - Socket – požadované rozhraní

Co je to komponenta?

- Dle specifikace UML 2.0 modulární část systému zapouzdřující svůj obsah a jejíž projev je nahraditelný v daném prostředí
  - Černá skřínka jejíž chování je plně definováno poskytovaným a požadovaným rozhraním
    - Nahraditelná jinou komponentou se stejným protokolem

Syntaxe komponenty

- Komponenty mají poskytována a požadována rozhraní, porty a vnitřní strukturu
  - Rozhraní delegují zprávy na vnořené části
  - Vnořené komponenty znázorněné bud uvnitř nebo vně a je použita relace závislosti.

Komponenty

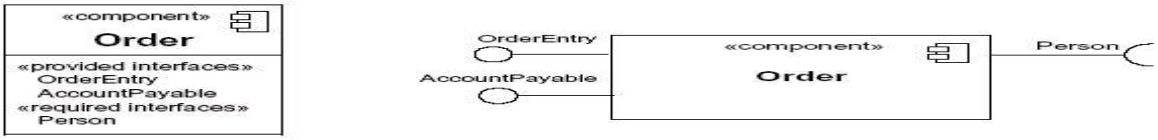
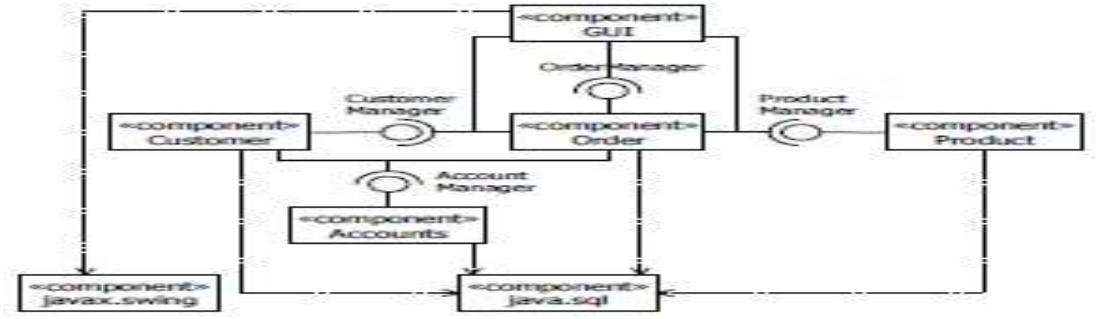
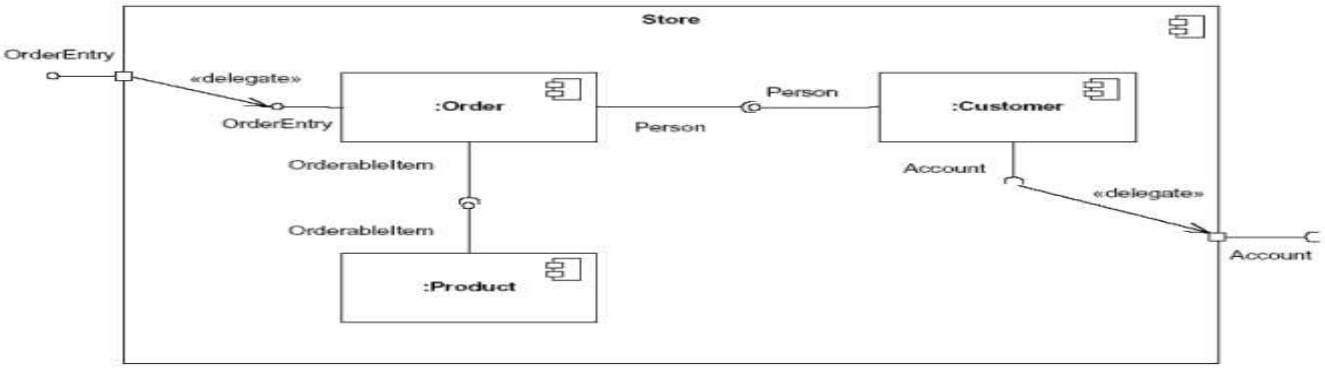
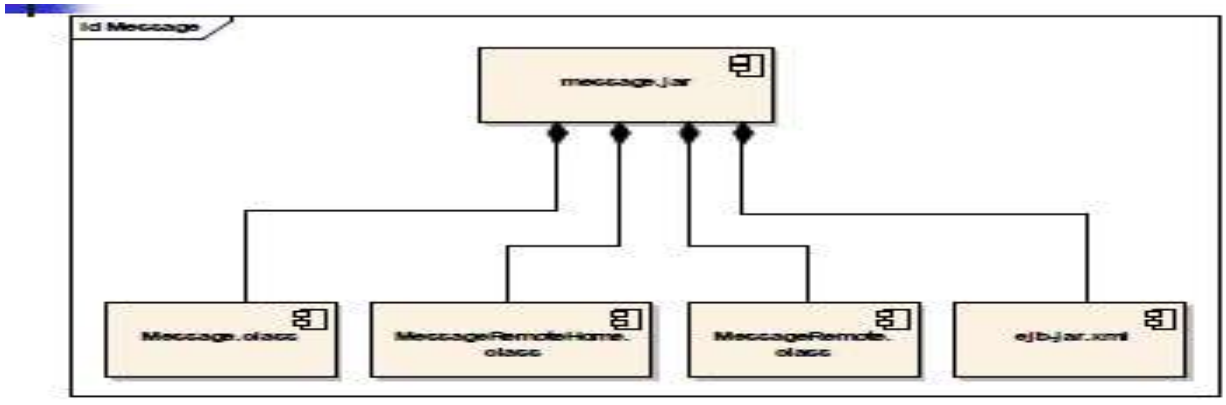


Diagram komponent

1. Vyjadřují fyzickou (vnitřní) strukturu komponent
   a. Přiřazení klasifikátorů (třídy, komponenty,…)
   b. Přiřazení artefaktů komponetám
2. Popisují systém jako kompozici komponent
   (závislosti mezi komponentami)





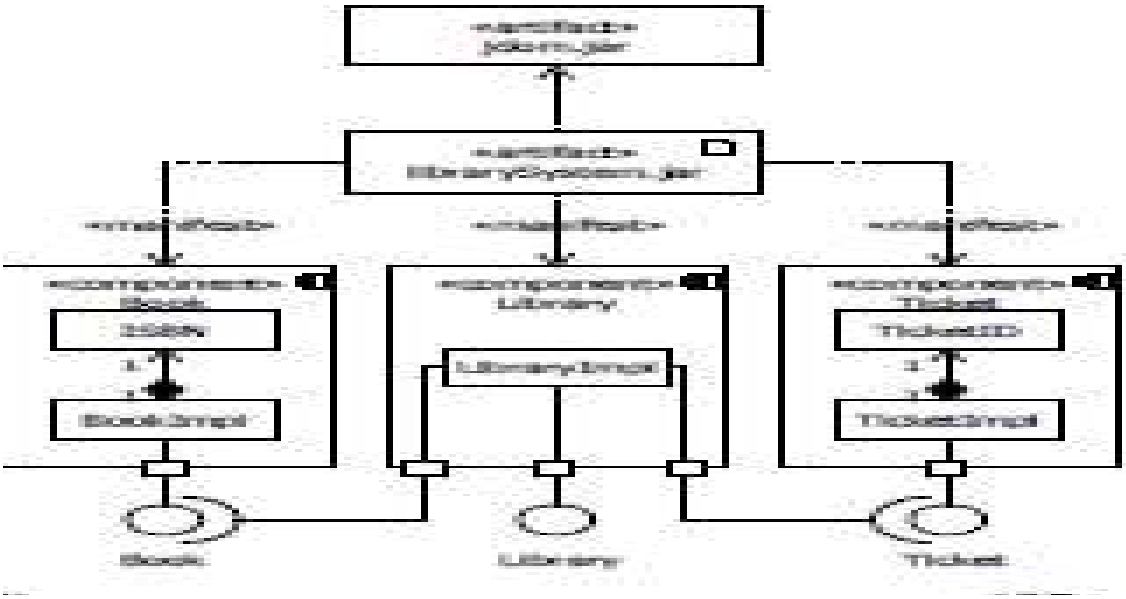


Artefakt

- Fyzickým projevem jedné či více komponent
  - Zdrojové soubory
  - Spustitelné soubory
    - Skripty
    - Dokumenty

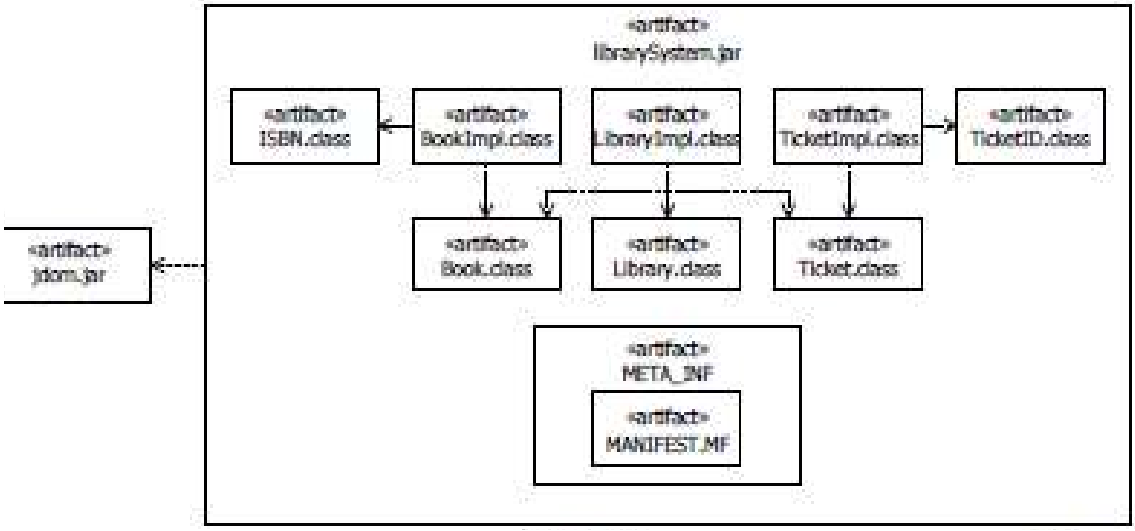
Artefakty a komponenty

- Artefakty jsou fyzickým projevem jedné či více komponent
  - Artefakt může obsahovat další artefakty
  - Artefakty mohou záviset na dalších artefaktech



Vztahy artefaktů

- Artefakty mohou záviset na jiných artefaktech – důsledek závislosti komponent



Standardní stereotypy artefaktů

Artifact stereotype	semantics
<<file>>	A fyzický soubor
<<deployment spec>>	Specifikace nasazení
<<executable>>	Dokument
<<library>>	Spustitelný soubor
<<library>>	Statická nebo dynamicky link, knihovna (DLL) nebo Java Archive (JAR)
<<script>>	Skript proveditelný pomocí interpretu

<<source>>	Zkompilovatelný zdrojový soubor

Stereotypy artefaktů

- Stereotypy definované profilem např. Java

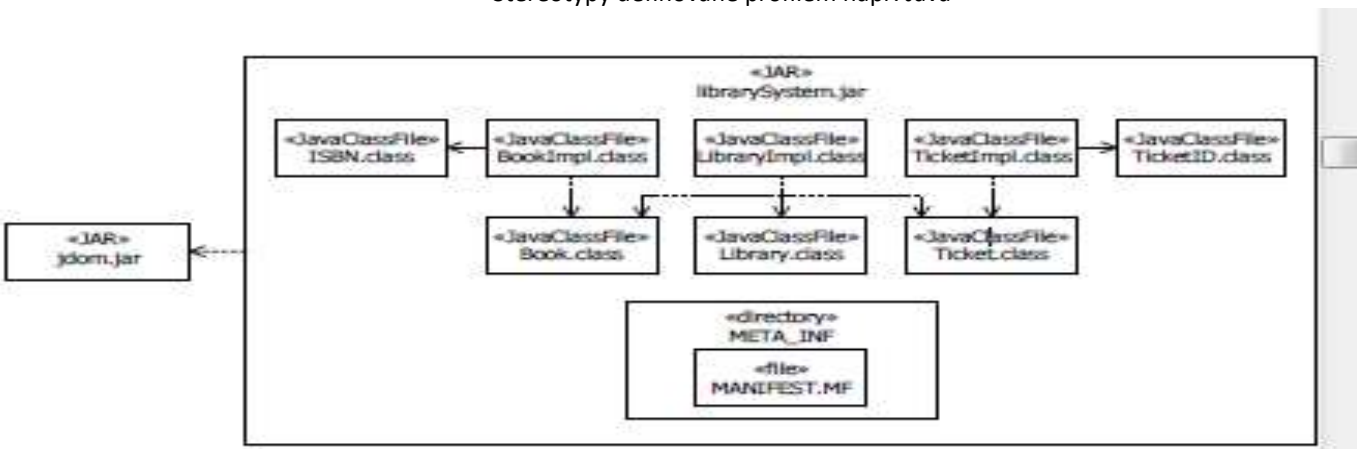


Diagram komponent

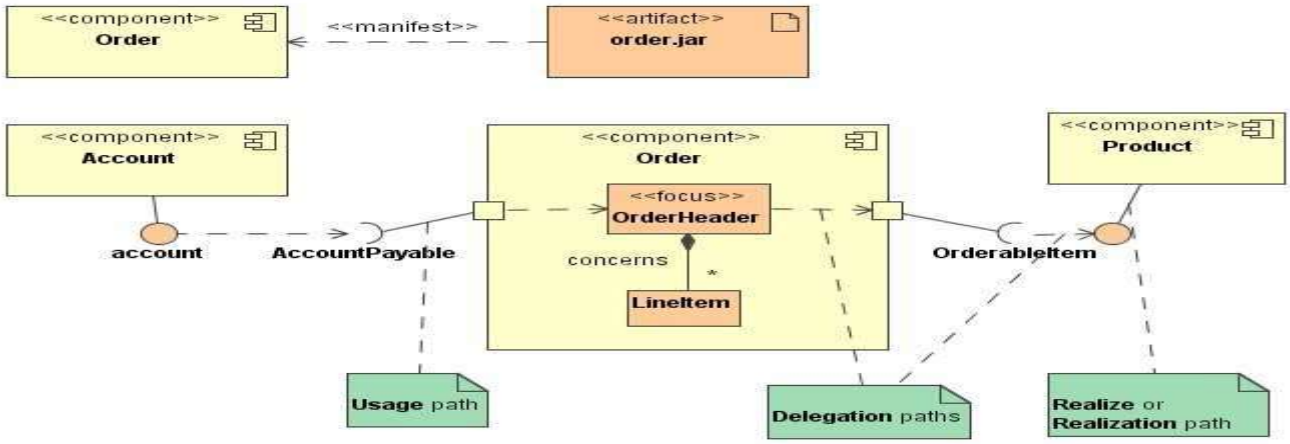
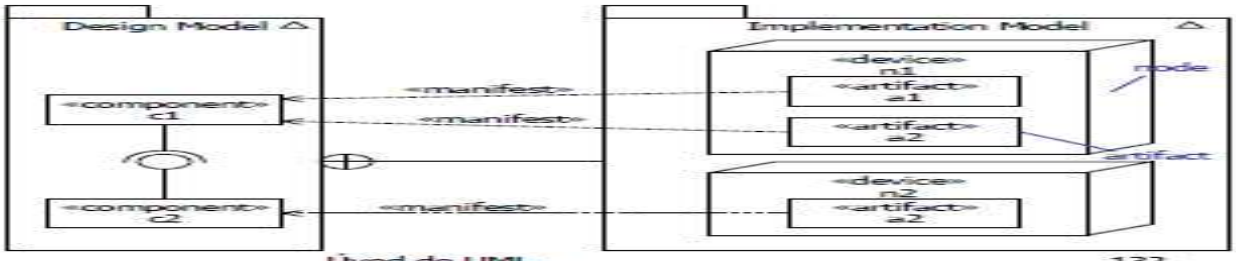


Diagram nasazení

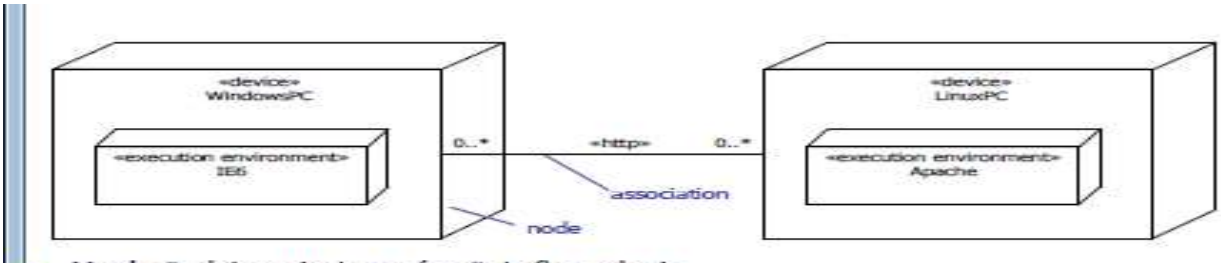
- Ukazuje přiřazení artefaktů uzlům resp.
- Ukazuje přiřazení instancí artefaktů instancím uzlů (diagram konkrétního nasazení)
  - Uzel
    - <<device>> - HW (Server, PC, pokladna
    - <<environement>> - SW (OS, www server, aplikační server,….)
  - Artefakt
    - Obvykle projevem jedné či více komponent
    - Fyzickým projevem libovolného prvku UML
      - Zdrojový soubory
      - Spustitelné soubory
        - Skripty
        - Dokumenty

Diagram nasazení vs komponent

- Komponent diagram – komponenty a realizující artefakty
- Deployment diagram artefakty nasazení na uzlech



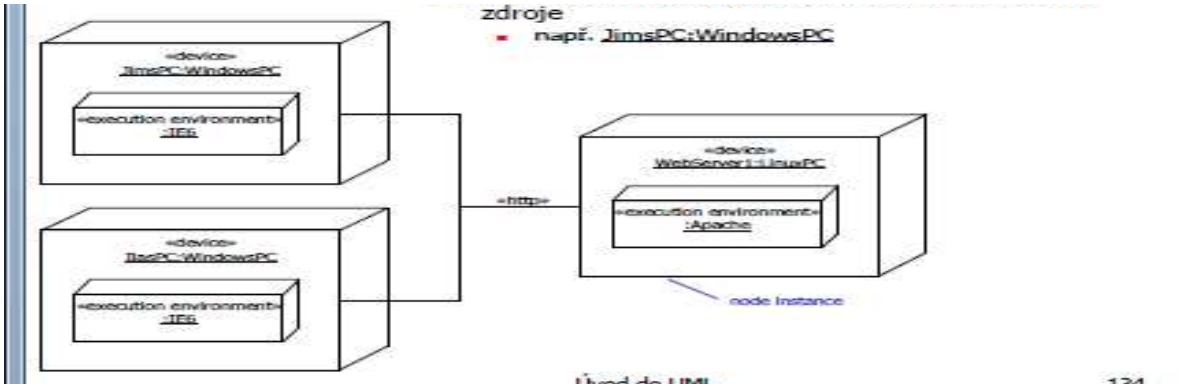
Uzly – typ



- Uzal představuje typ výpočetního zdroje
  - Např. a WinodwsPC

Uzly – instance

- Uzel reprezentuje konkretní instanci výpočetního zdroje
  - Např. JimsPC:WindowsPC



Nasazení

- Artefakty nasazení na uzly (typy), instance artefaktů nasazení na instance uzlů

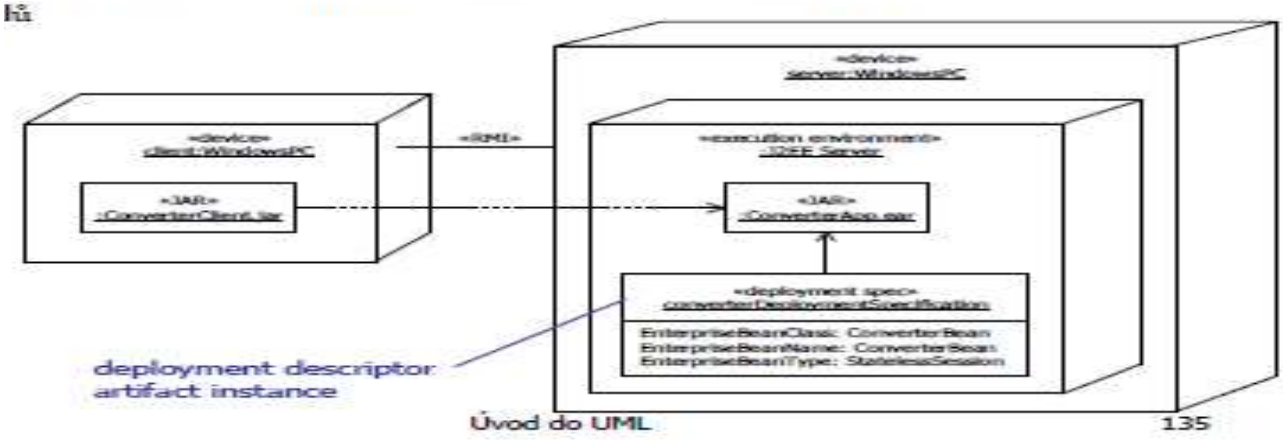


Diagram nasazení

- Hlavní prvky . uzel, komponenty, artefakt
- Hranice mezi použití ikon pro uzel, artefakt, komponentu – neostrá

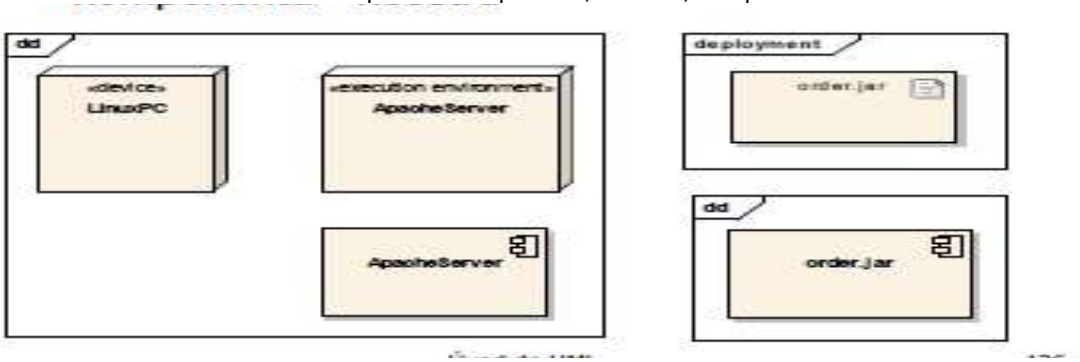
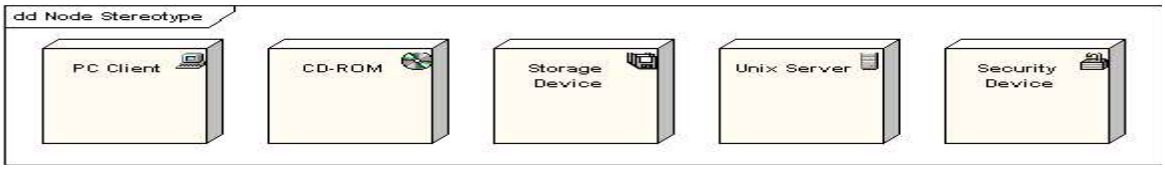
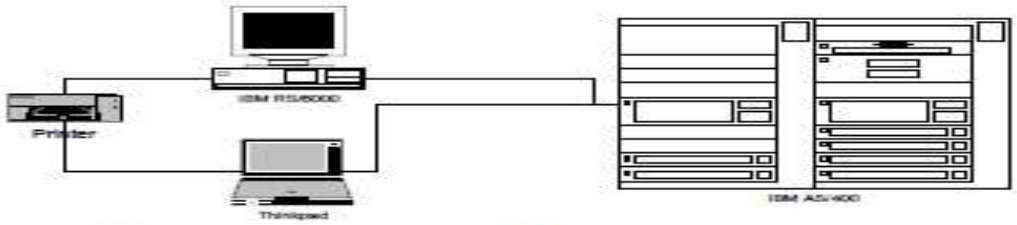


Diagram nasazení

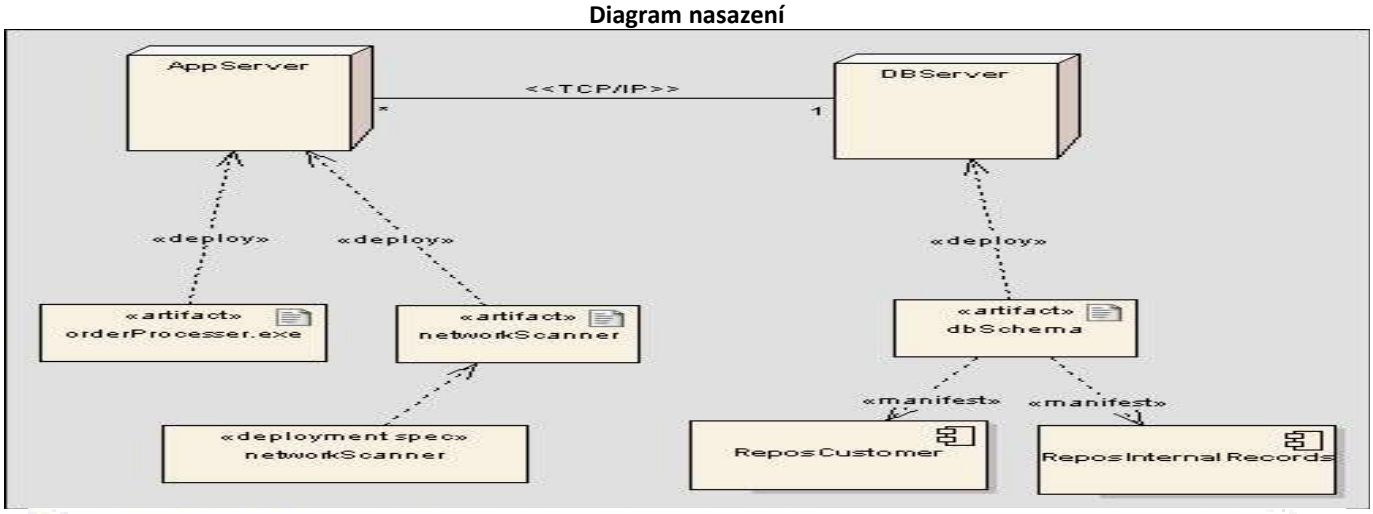


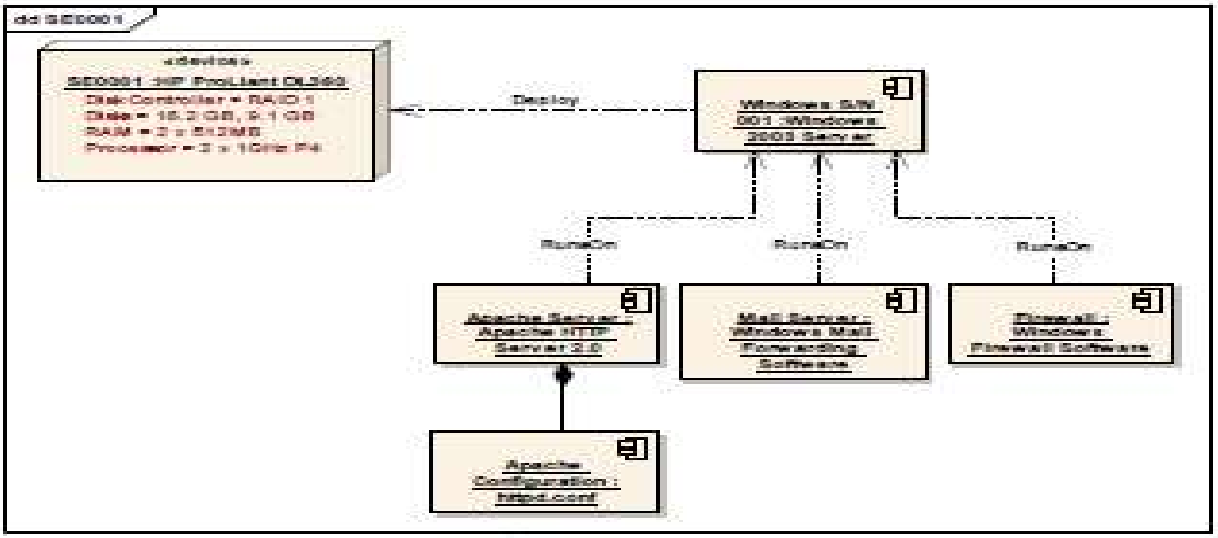
Vlastní ikony



- Ikona zastupuje uzel s určitým stereotypem

Diagram nasazení





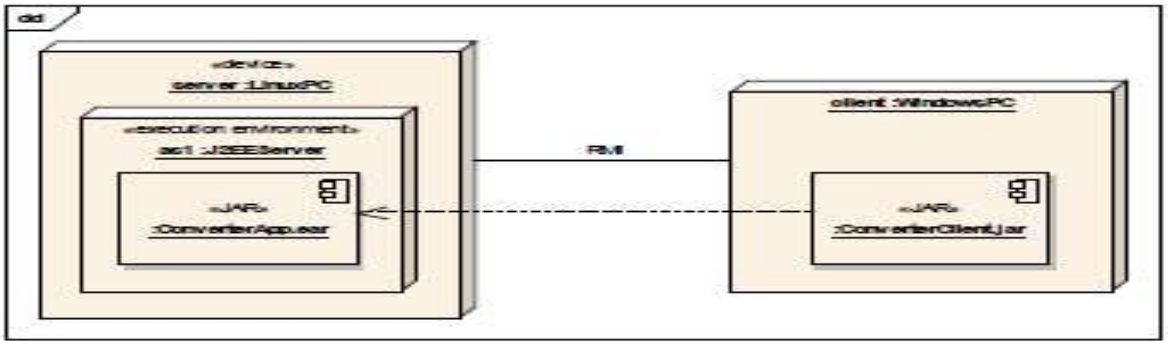
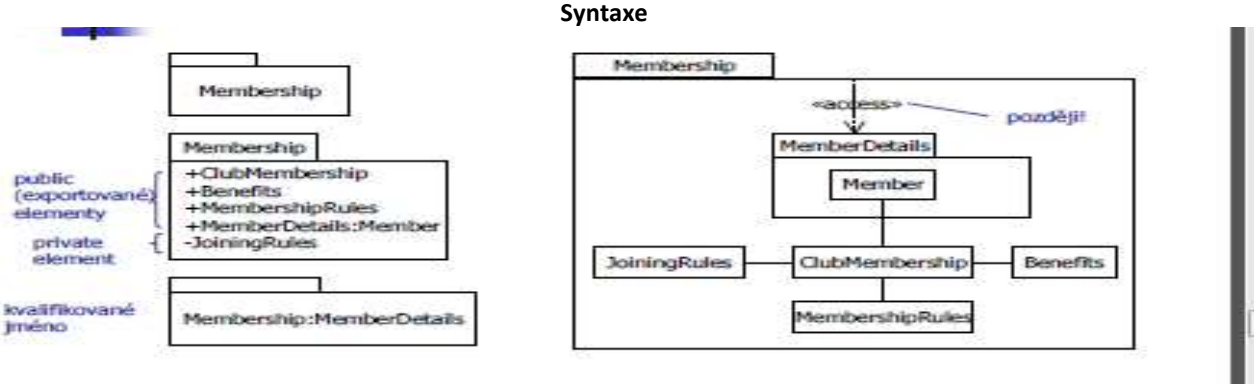


Diagram balíčku

Balíček

- Mechanismu pro spořádání prvků a diagramů do skupin
  - Účel
    - Mechanismus entity velkých modelů
    - Jednotku pro souběžnou práci
    - Jednotky pro správu konfigurace
    - Poskytující zapouzdření jmenné prostory
  - Doporučení
    - Seskupit sémanticky příbuzné elementy
    - Minimalizace relace napříč balíčky

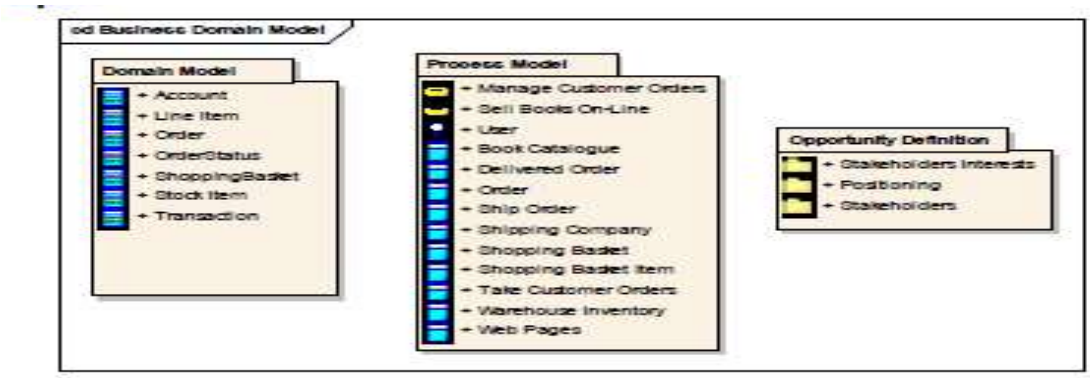
Syntaxe



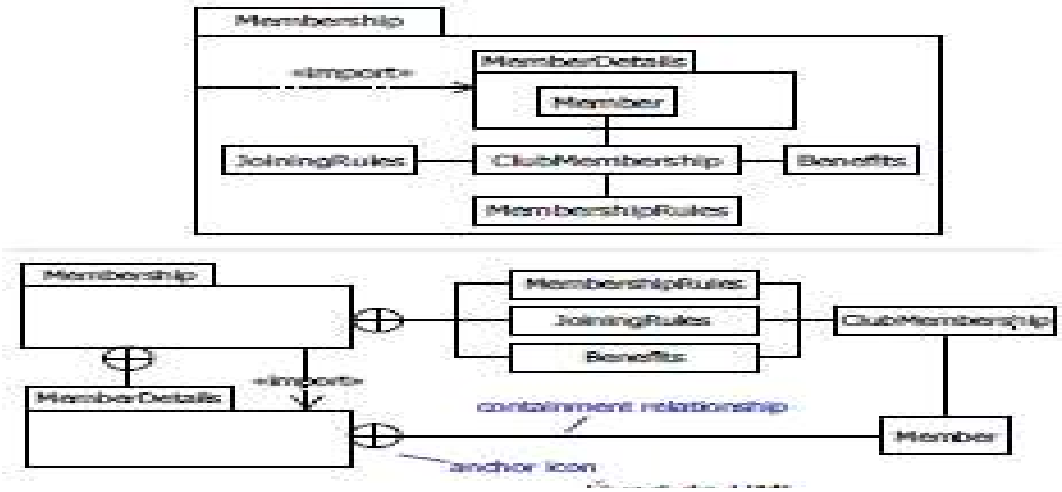
Viditelnost prvků v balíčku

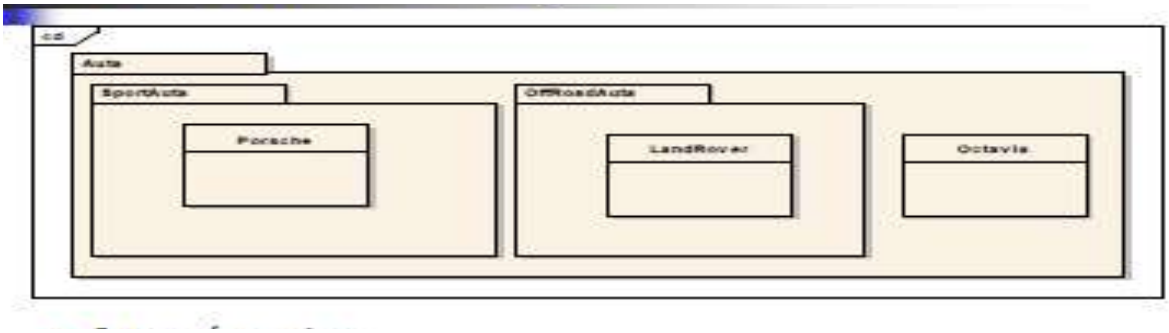
- Viditelnost prvků realativní vzhledem k jeho balíčku
  - \+ public – viditelný všem i mimo balíček
  - # protected – viditelný zděděným balíčků
  - – private – nevidetlný mimo balíček

Balíček

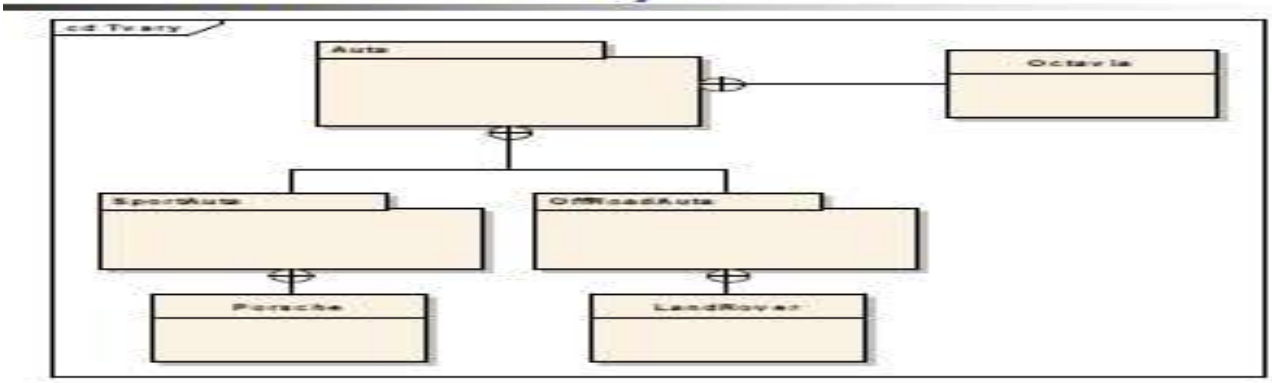


Vnořené balíčky



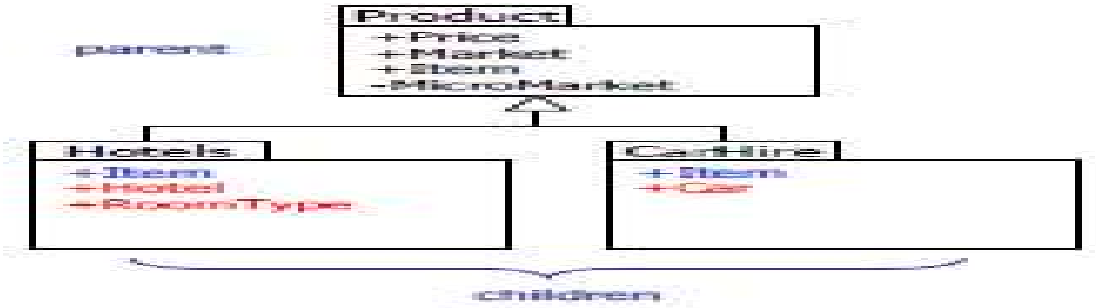


- Jmenné prostory
  - Auta::SportAuta::Porsche
  - Atuo::OffRoadAuta::LandRover
  - Auta::Octavia
- V rámci balíčku – bez prefixu

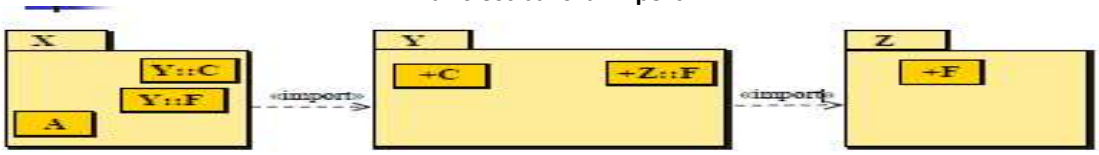


Generalizace balíčků

- Potomek dědí veřejné a chráněné prvky
- Potomci mohou přetížit prvky z předka
- Potomci mohou přidat nové prvky

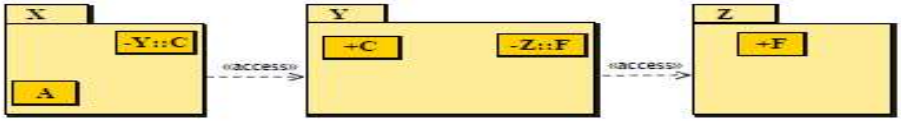


Závislost balíčků-import



- Veřejně importovaného balíčku se stávají veřejnými prvky importujícího

Závislost balíčků – access

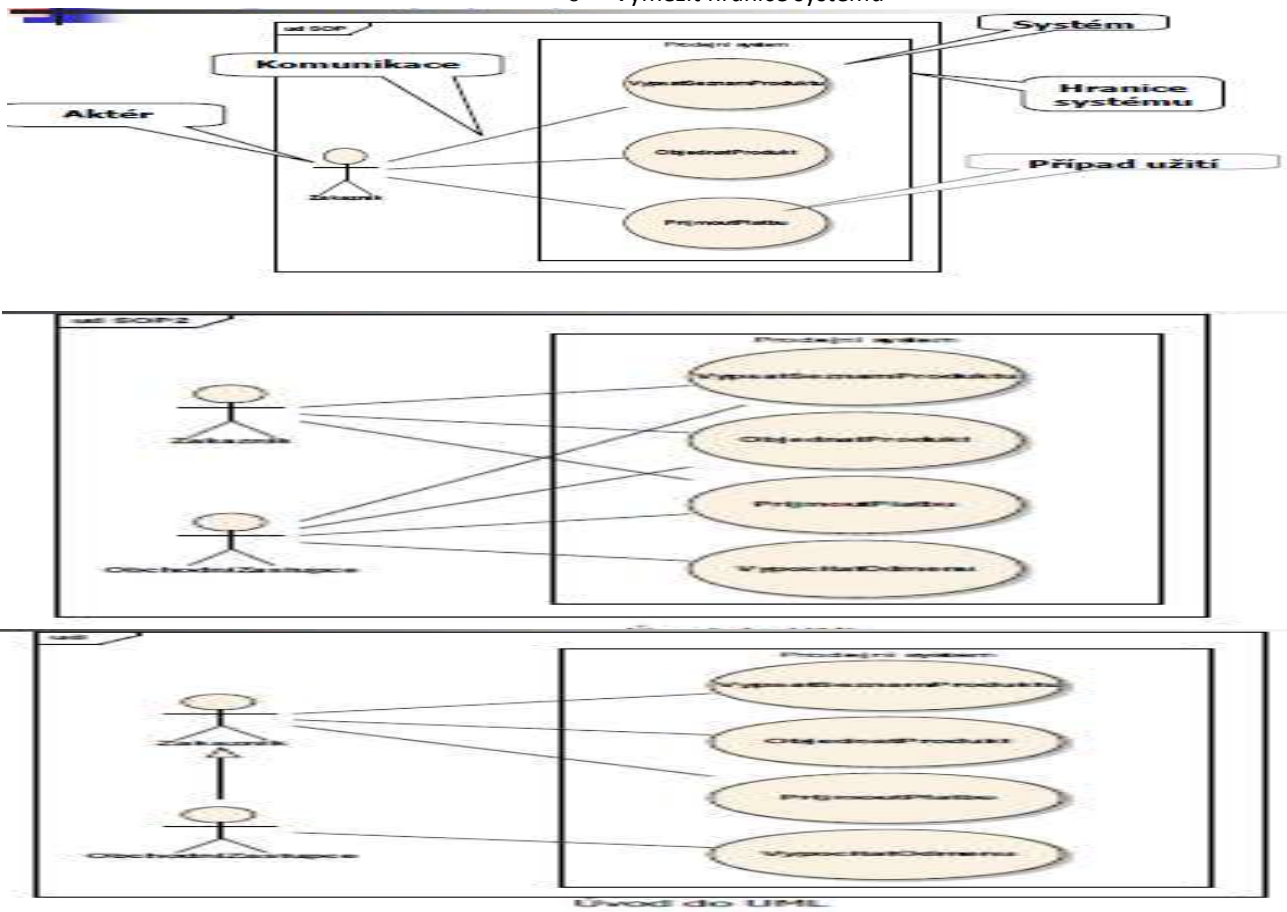


- Veřejné prvky importovaného balíčku se stávají soukromými prvky importujícího

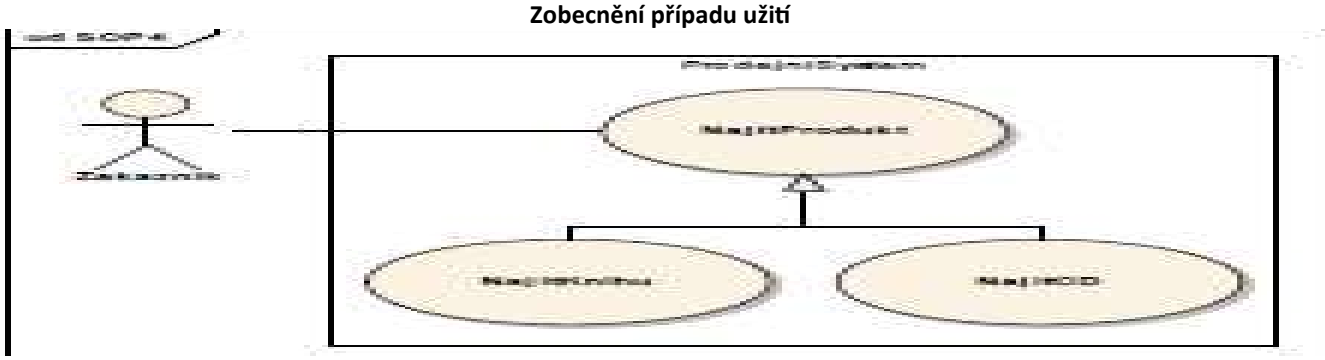
Diagram případu užití

- Zobrazují, kdo a jakým způsobem užívá navrhovaný systém
- Cíle tvorbu DPU
  - Najít aktéry-uživatele systémy

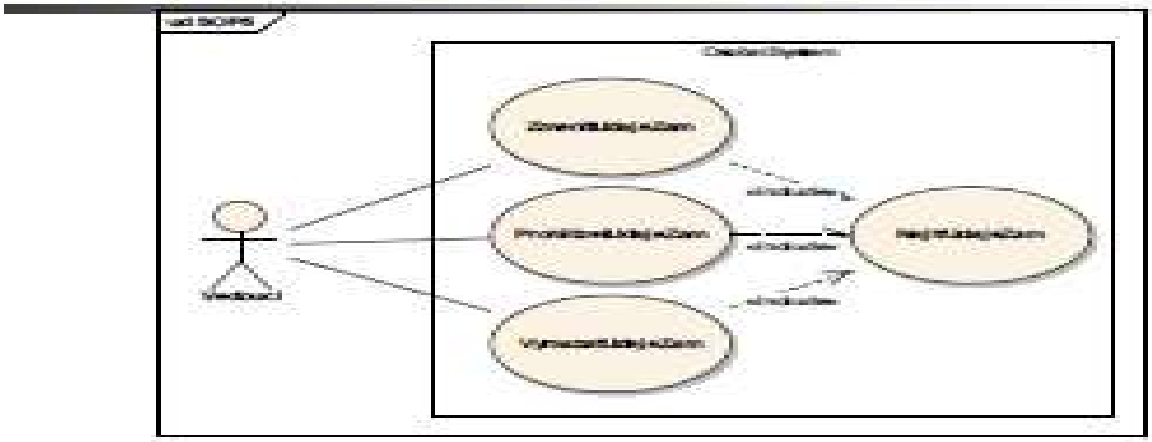
- Zjistit jak systém používají
- Vymezit hranice systému



Zobecnění případu užití

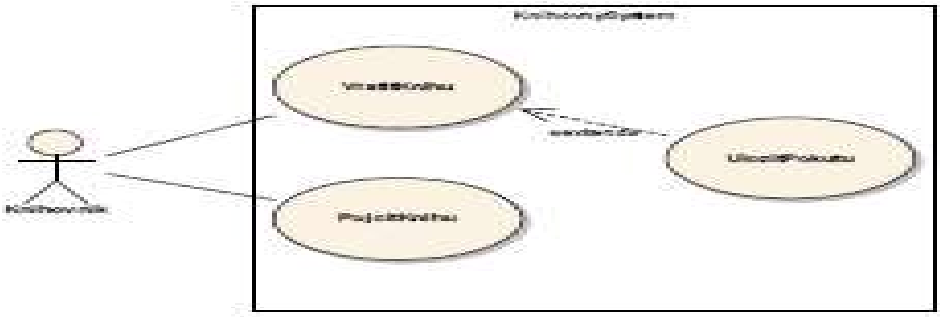


Relace <<include>>



- Vkládaný prvek je nutný pro existence vkládajícího
- Vkládáný prvek obvykle opakujícíc se rutina

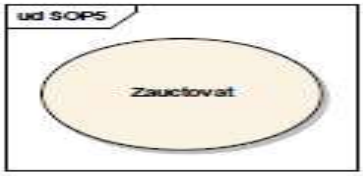
Relace <<extend>>



- Vkládaný prvek je volitelný
- Vkládající prvek je úplný

Model jednání

- Model jednání =
  - Diagram připadu užití +
  - Slovní scénář pro každý případ užití



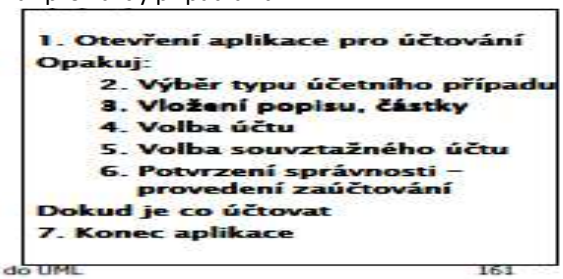


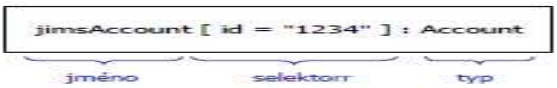
Diagram interakce

- Realizace USE CASů jako interakce instance klasifikátoru
- Sekvenční diagram
- Diagram komunikace
- Přehledové diagram
- Časové diagram

Hlavní prvky diagramů interakce

- Čáry života (Lifelines)
- Zprávy – komunikace mezi lifelines

Čáry života (Lifelines)



- Jméno – the name used to refer to the lifeline in the interaction
- Selector – logická podmínka určující konkrétní instance
- Typ

Musí být unikátní v rámci diagram

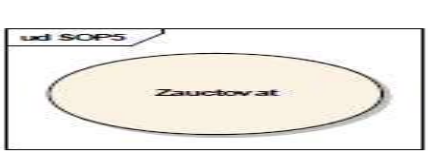
Stejná ikona jako klasifikátor, který ji reprezentuje

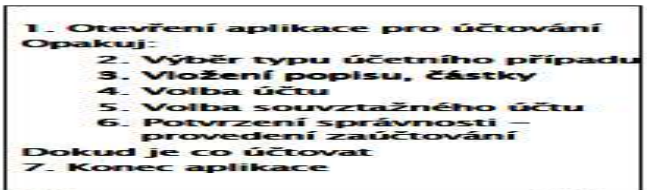
Zprávy

sender → receiver/target	typ	sémantika
→	synchronní	odesílatel čeká na odpověď
→	asynchronní	odesílatel nečeká na odpověď
←- - - -	návrat	returning from a synchronous operation call návrat ze synchronního volání
- - - →:A	konstrukce	odesílatel vyváří cíl
→ ✕	destrukce	odesílatel destruuje cíl
● →	nalezená zpráva	the message is sent from outside the scope of the interaction
→ ●	ztracená zpráva	the message fails to reach its destination

Sekvenční diagramy

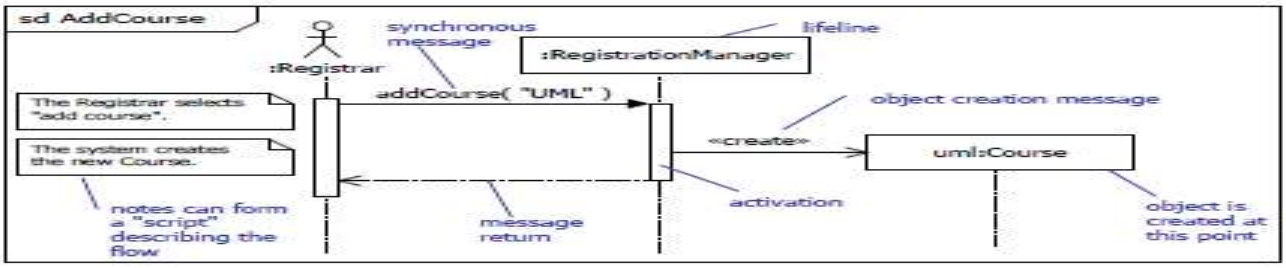
- Realizace případů užití jako spolupráce komunikacích instancí tříd



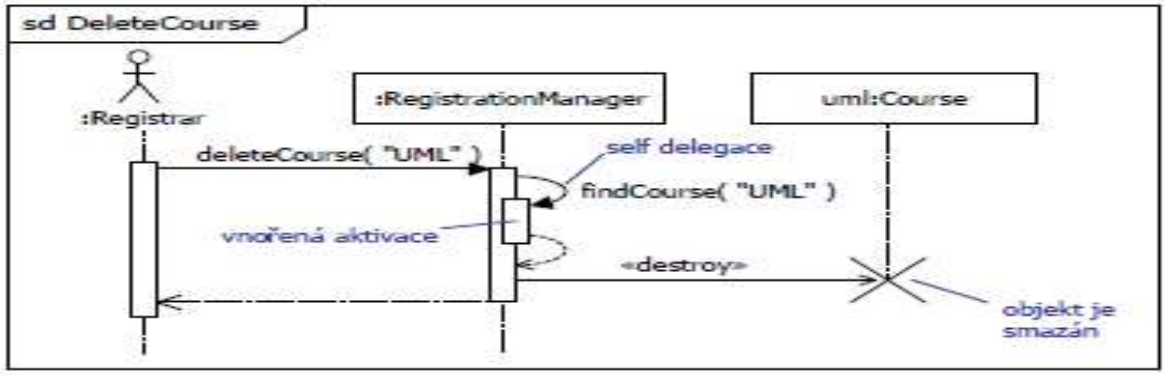


Sekvenční diagramy

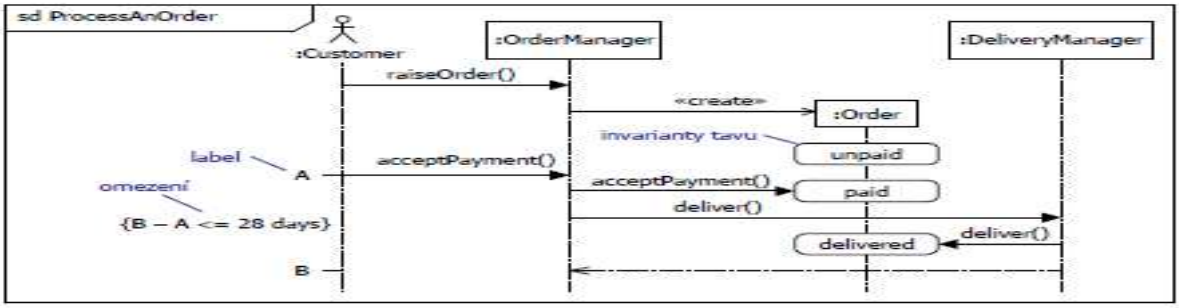
- Vstupem pro tvorbu
  - Diagram případu užití
    - Slovní scénář
    - Diagram tříd
- Během tvorby dochází k vytvoření
  - Nových metod
  - Nových metod tříd
- Kladou důraz na časové hledisko
  - Ukazují
    - Kdo (instance tříd) se podílí na realizaci
    - Jaké zprávy a s jakými parametry si objekty vyměňují
    - V jakém pořadí



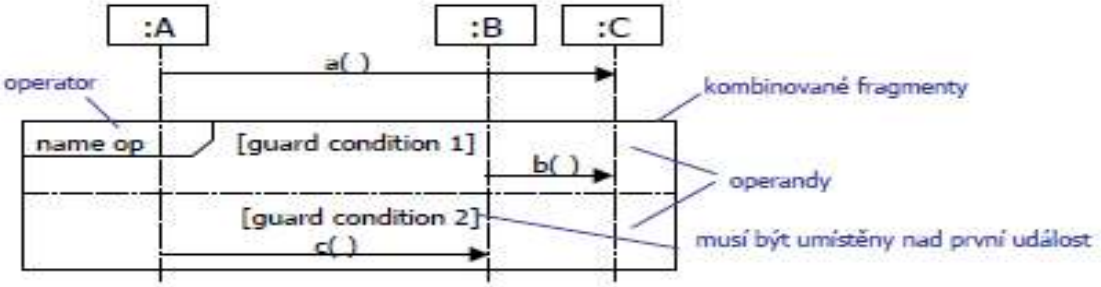
- Aktivace nejsou povinné



Invarianty stavu a omezení

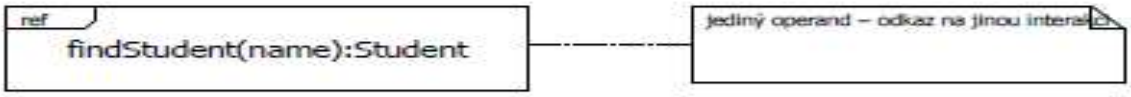


Kombinované fragmenty



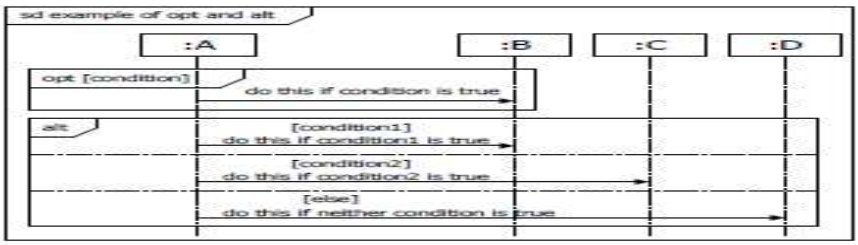
- Operátor říká jak (režim) budou operandy spušteny
- Guard condition – podmínka spuštění daného operandu/ů

operator	long name	semantika
opt	Option	if ...then...
alt	Alternatives	switch...case...
loop	Loop	loop min, max [podmínka] Iteruje minimálně *min times*, pokračuje do *max times* dokud podmínka splněna
break	Break	zbytek je přeskočen
ref	Reference	odkaz na jinou interakci= volání procedury



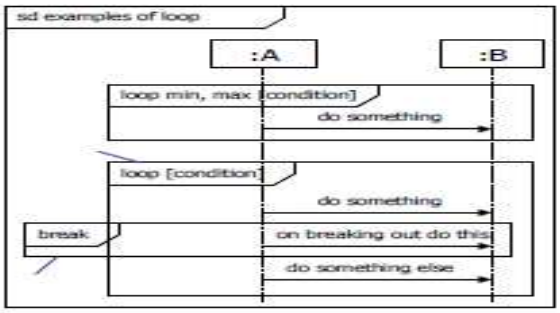
Větvení pomocí opt a alt

- opt :
  - pouze 1 operand
- alt :
  - 2 a více operandů

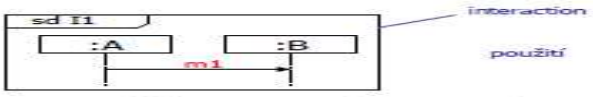


Iterace s loop a break

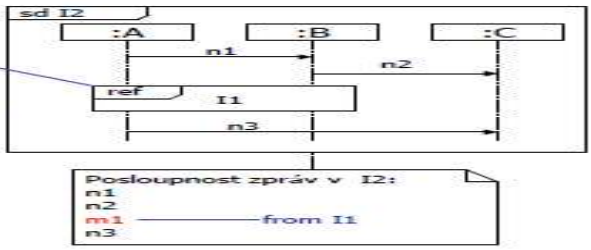
- loop sémantika:
  - dělej min krát, potom dělej (max – min) krát dokud podmínka platí
- Break –
  - provede se při přerušení cyklu
  - zbytek cyklu se nespustí
- break je vně cyklu



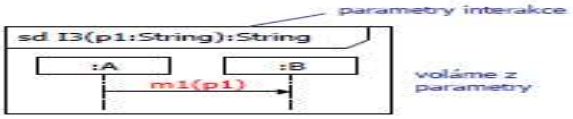
Volání jiné interakce

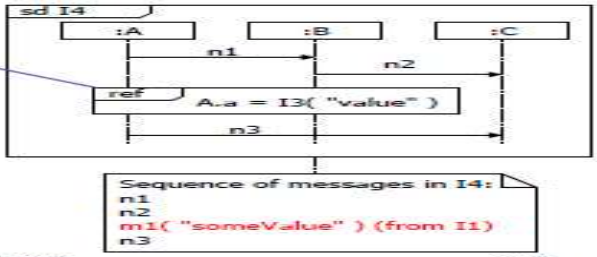


- Použité lifelines musí být i ve volajícím sd
- Volání protíná lifelines, které používá

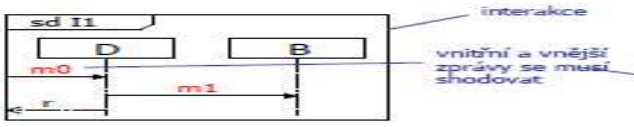


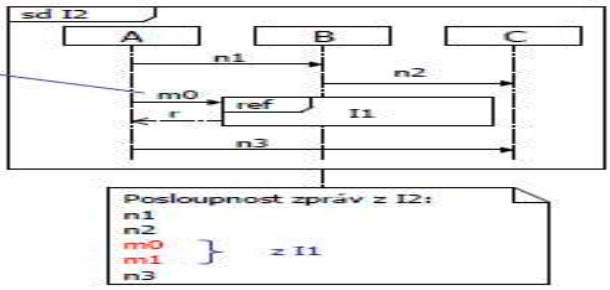
Parametry





Brány





Body pokračování

- první neb poslední prvek bloku
- zanechá instance v takovém stavu, aby se dal navázat
- musí obsahovat stjné lifelines

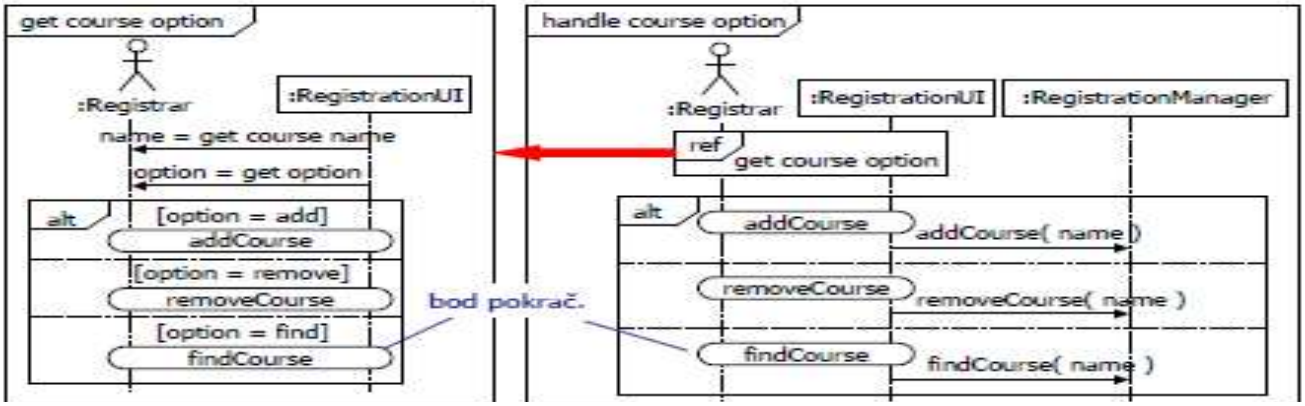
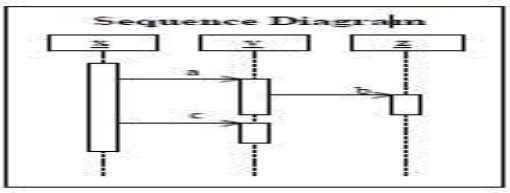
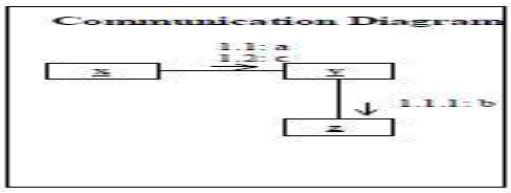


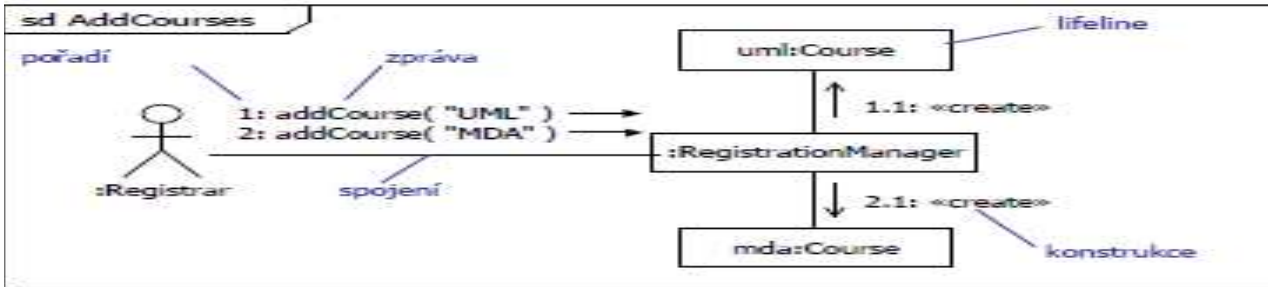
Diagram komunikace

- Stejný účel jako sekvenční diagramy
  - Zdůrazňují strukturní hledisko
    - Jakým způsobem jsou objekty při spolupráci přivázané
      - Čas zachycen pomocí číslování zpráv

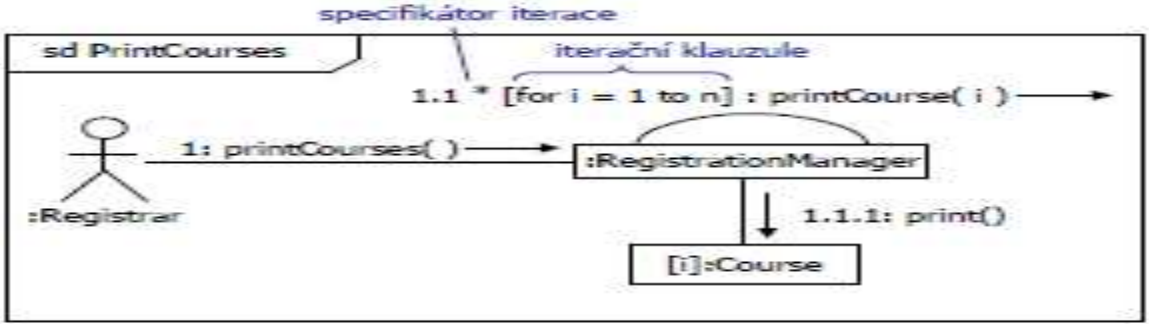




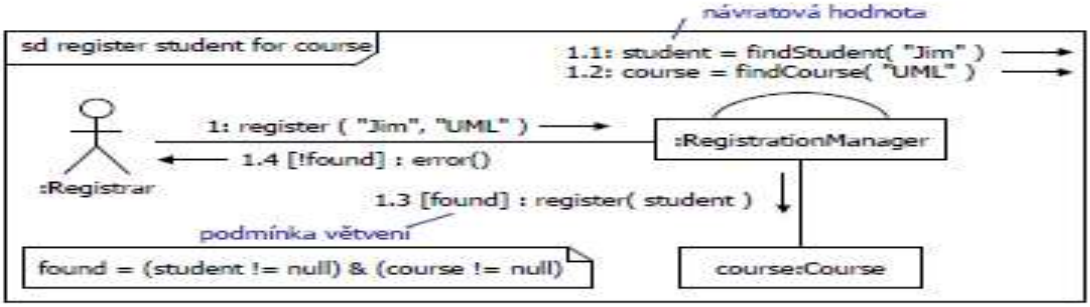
Syntaxe



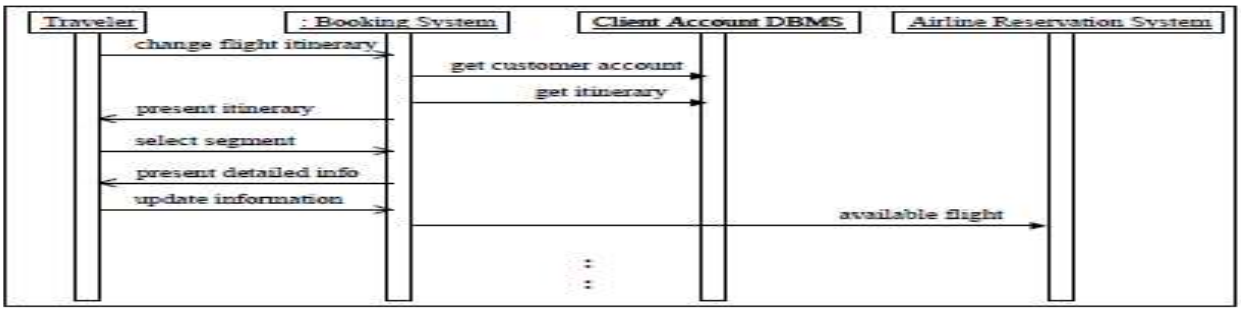
Iterace



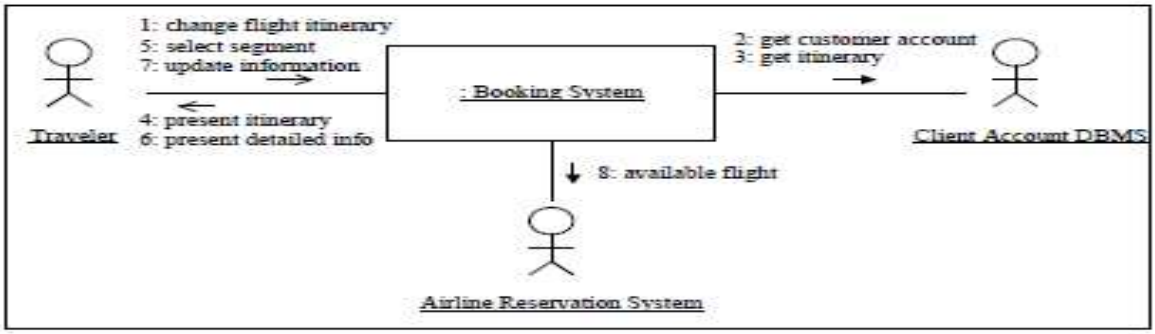
Větvení



Diagramy komunikace

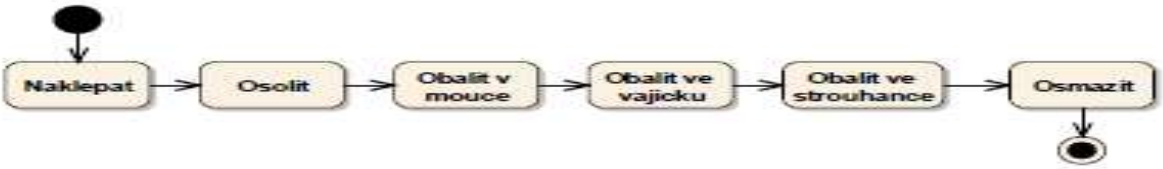


Diagramy komunikace



Diagramy aktivit

- Objektově orientované vývojové diagramy
  - Popis procesu
  - Proces složen z dílčích subprocesů



- Založené na Petriho sítích
  - Hra s tokeny

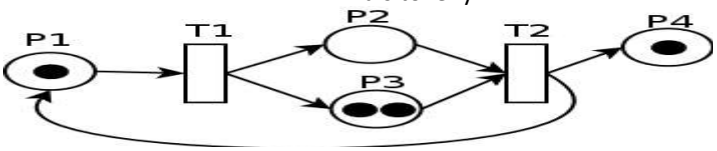
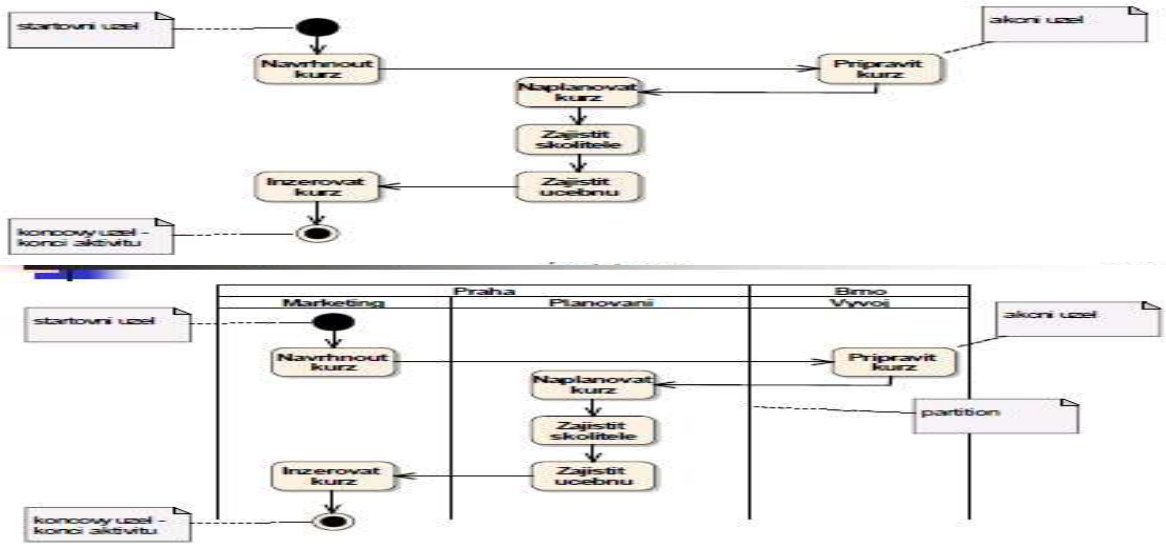


Diagram aktivit

- Nejčastěji použité
  - Analýza
    - Průchod USE Casey
    - Scénář z procesního hlediska
  - Návrh
    - Popis organizace
  - Modelování organizace
    - BMP- modelování firemních procesů



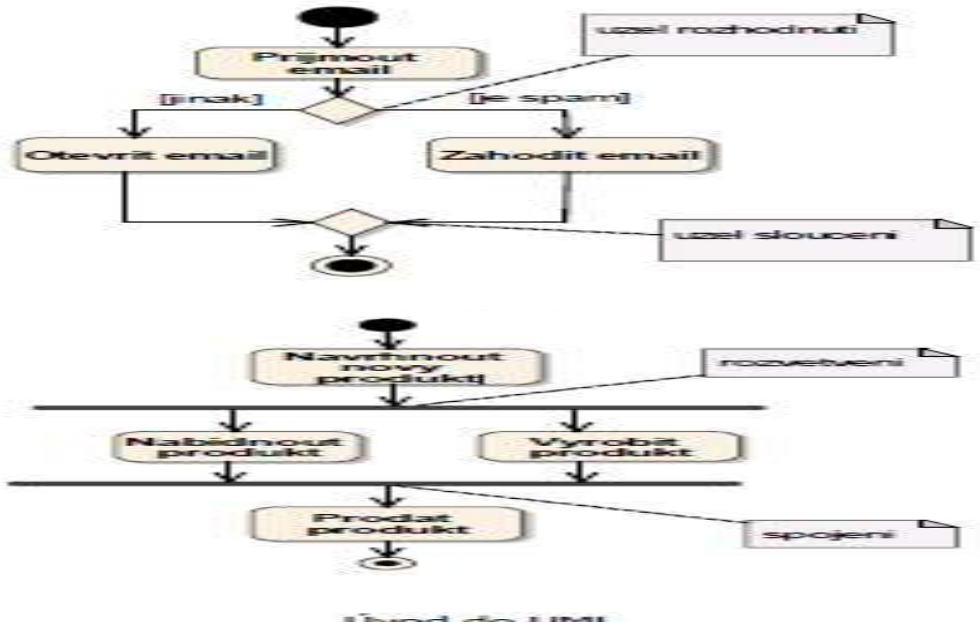
Akční uzel

- Volání akce
- Poslání signálu
- Čekání na signál

Akční uzel – volání akce

- Spuštěn – tokeny na všech vstupních hranách AND
- Po dokončení akce – tokeny na všech výstupních hranách

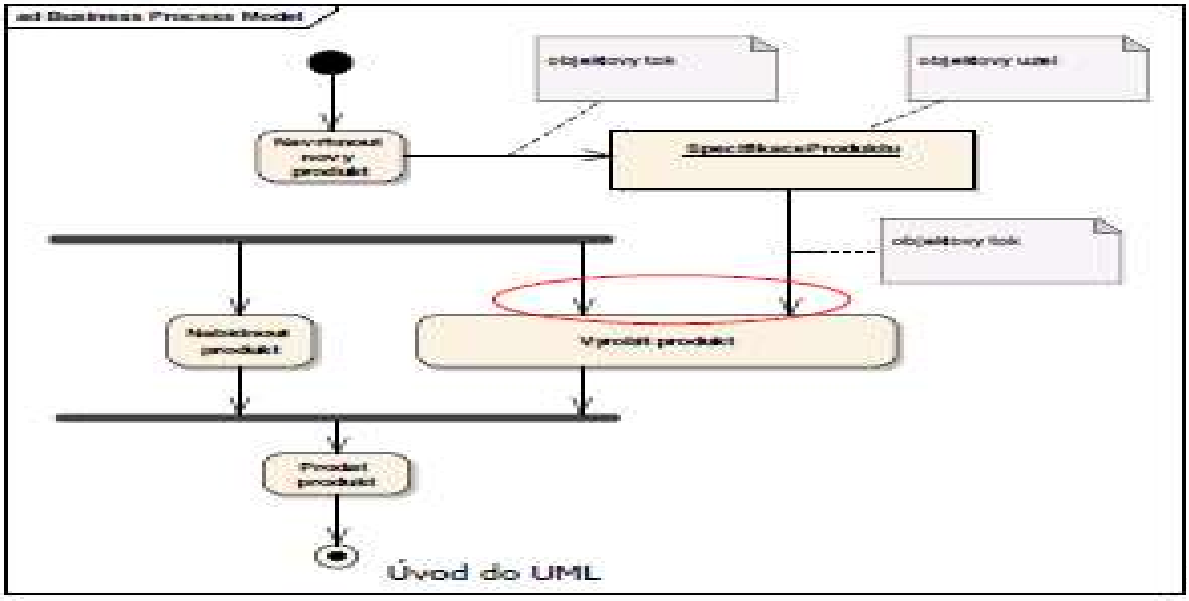
Diagram aktivit



Řídící uzel

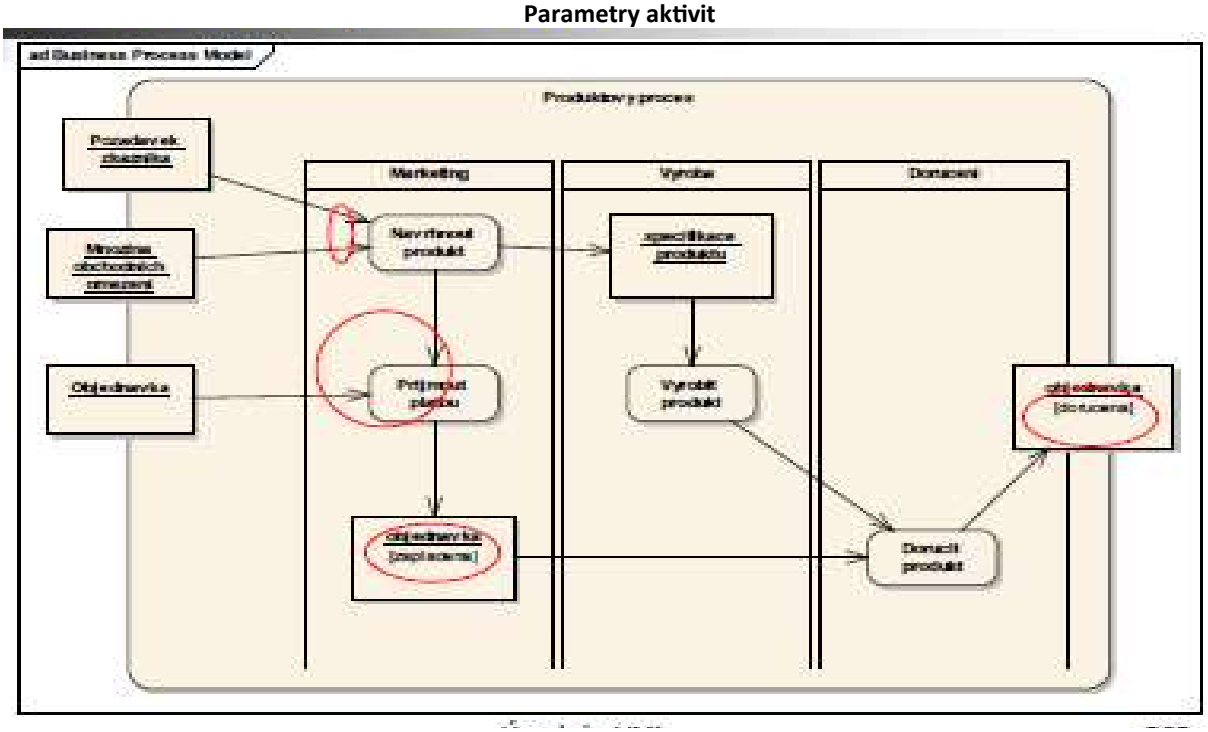
- typy
  - startovní (může být více)
  - koncový aktivity
  - koncový cesty (aktivita nekončí)
  - rozhodnutí (větvení, max. 1 aktivní)
  - sloučení
  - rozvětvení (paralel. běh)
  - spojení

Objektový uzel

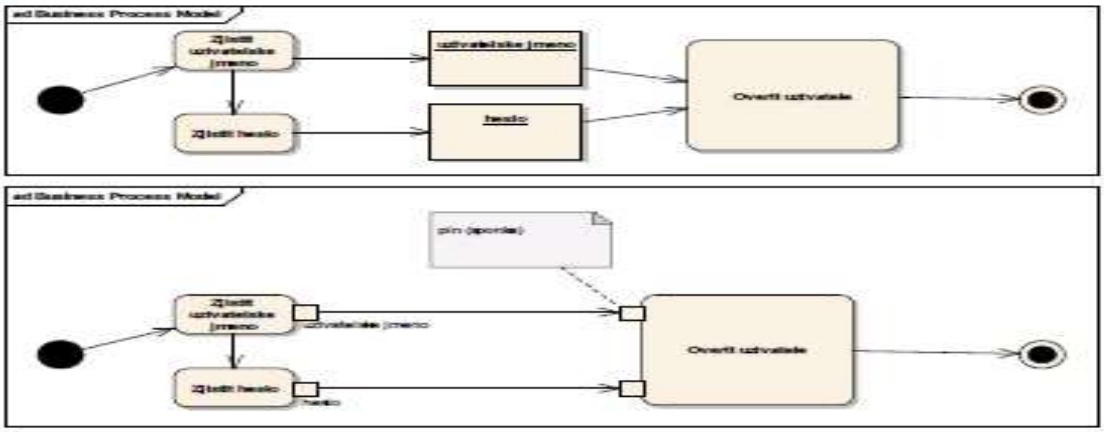


- Vyrovnávací paměť
- Takone čeká v uzlu, dokud není příjmut na nějakém výstupu
- Výstupy soupeří – token se neduplikuje

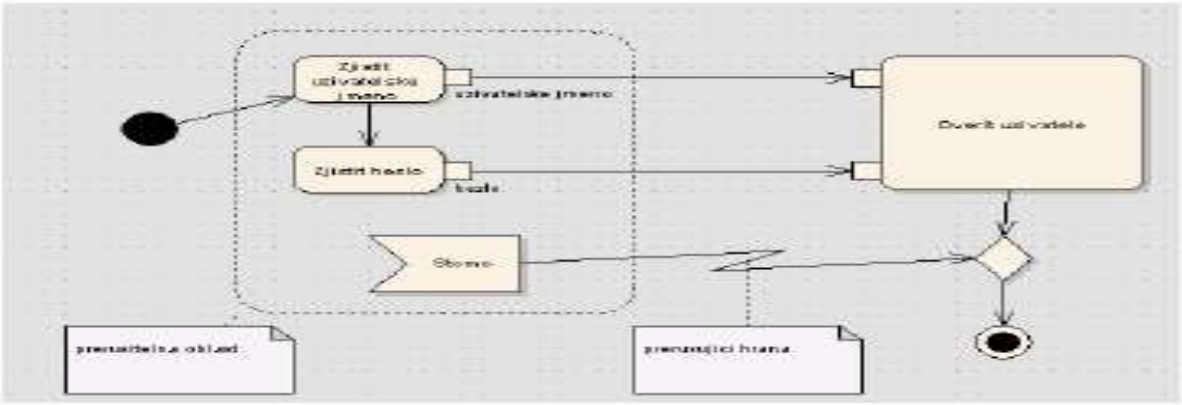
Parametry aktivit



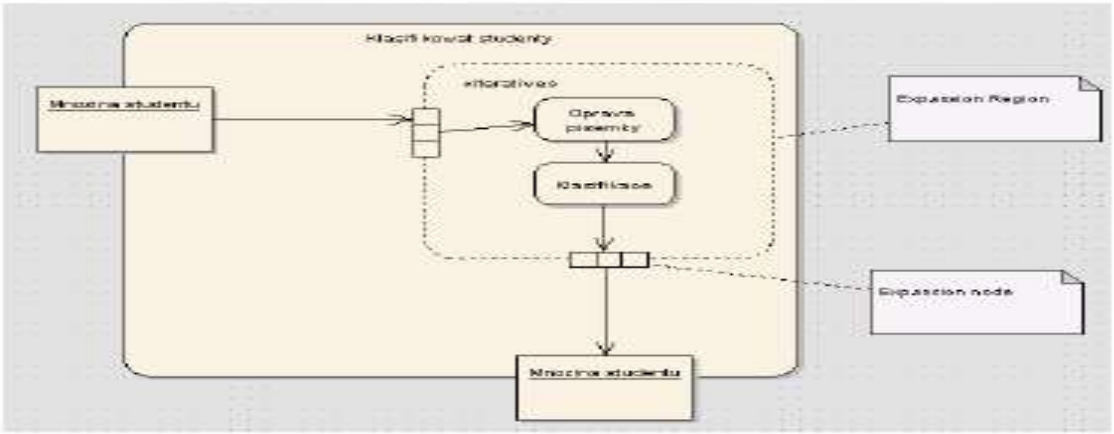
Sponka (Pin)



Přerušitelná oblast



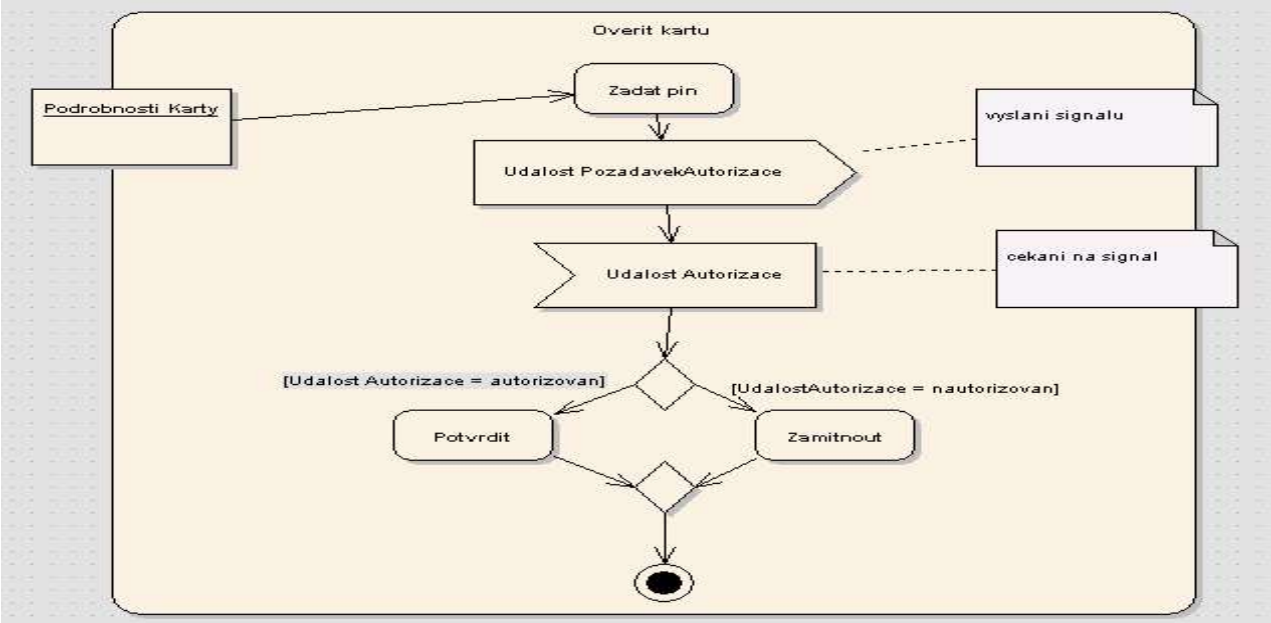
Přídavné uzly



Přídavné uzly

- Zpracování kolekcí
  - Módy
    - Iterativní
    - Paralelní
    - Proudový

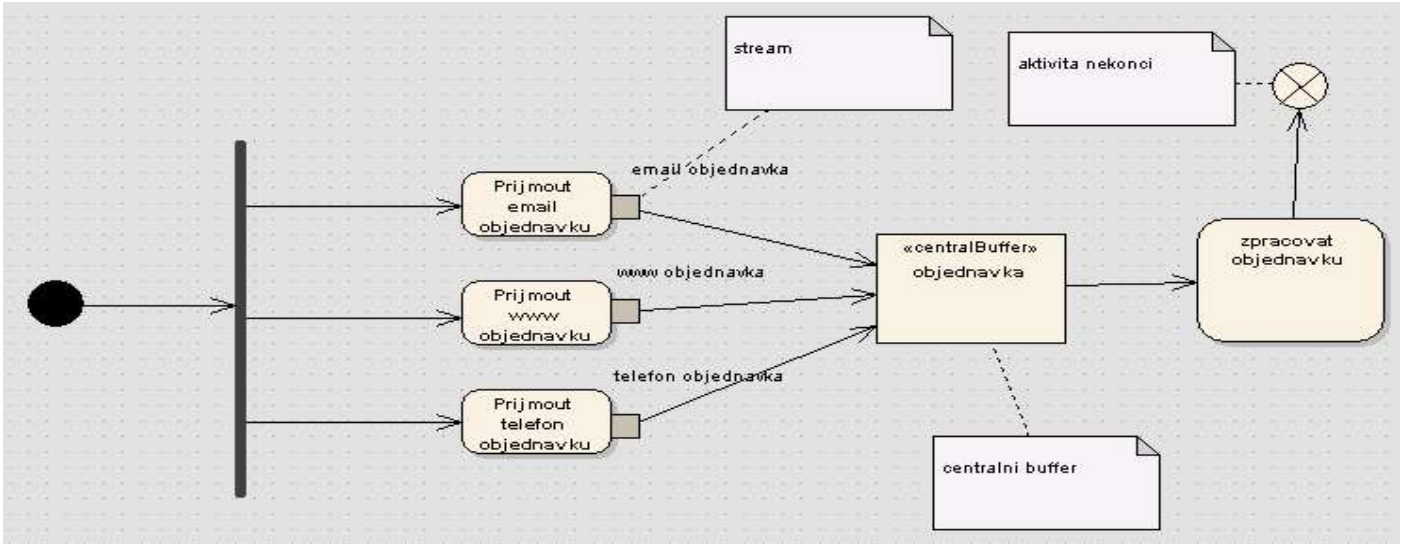
Signálové uzly



Signálové uzly

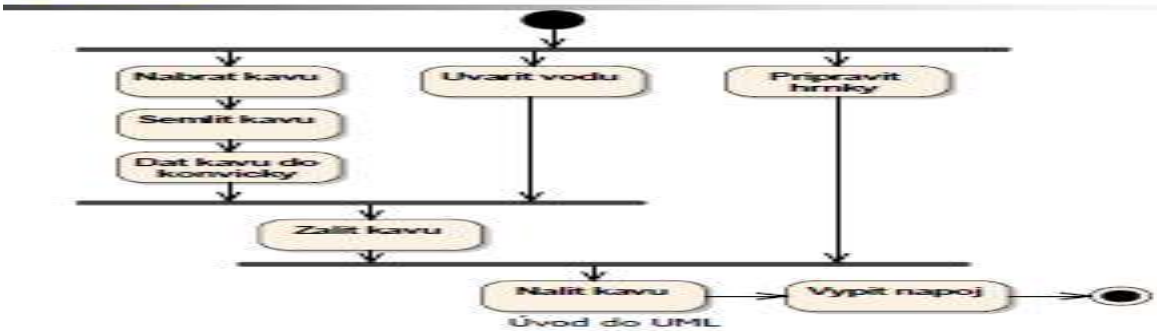
- Speciální typ akčních uzlů
  - Vyslání
    - Asynchronní (nečeká se na odpověď)
      - Více vstupů (AND)
  - Přijetí
    - Příjme se token a čeká se na signál
      - Max. jeden vstup
        - Jeden výstup

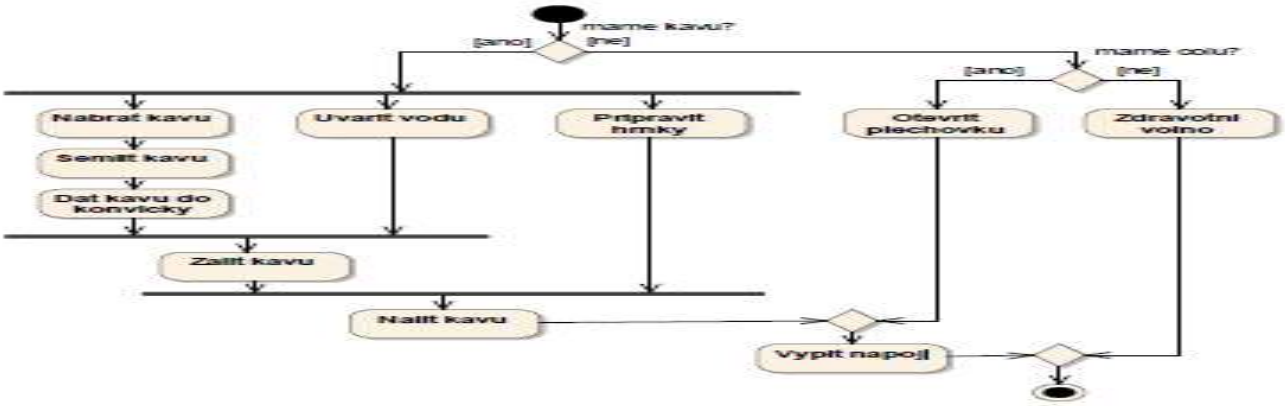
Centrální buffer



- Převod proudu na iteraci

Příklad – doplnění kofeinu





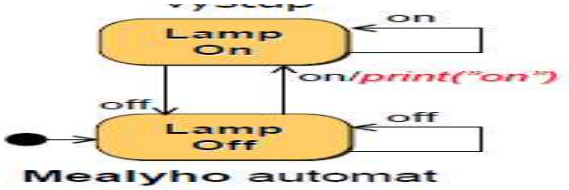
Stavové automaty

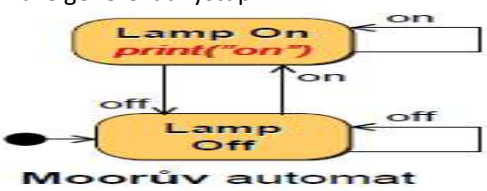
Stavový diagram

- 1 stavový automat pro 1 reaktivní objekt
  - Reaktivní objekt
    - Reaguje a vnější události
  - Životní cyklus jako řada stavů, přechodů a událostí
  - Jejich chování důsledkem předchozího chování
- SA popisuje životní cyklus reaktivní objekt
  - Reaktivní objekt např.
    - Třída
    - Systém
    - Podsystém

Výstupy a akce

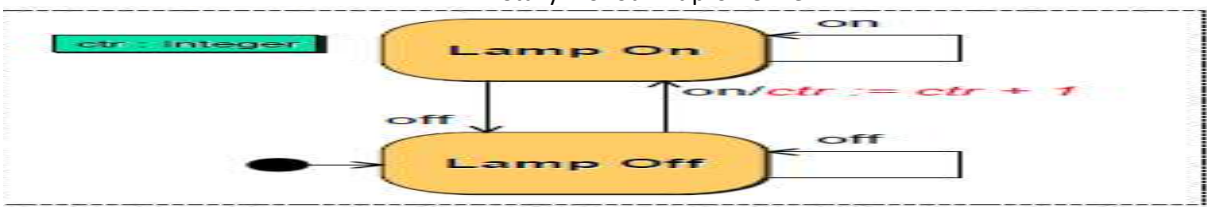
- Při změně stavu automat může generovat výstup





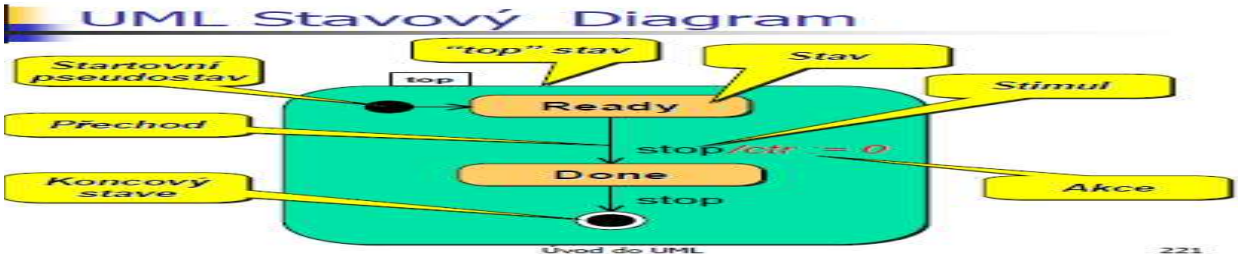
Rozšířené stavové automaty

- Stavy mohou mít proměnné



Trochu teorie

- Rozšíření Mealyho stroj je definován:
  - Množinou vstupních signálů (stimulů) (vstupní abeceda)
  - Množinou výstupních signálů (výstupní abeceda)
  - Množinou stavů
  - Množinou přechodu
    - Odkud
    - Kam
    - Stimul
    - Akce
  - Množinu proměnných stavu
  - Počátečním stavem
  - Množinou koncových stavů (pokud se uvažuje)

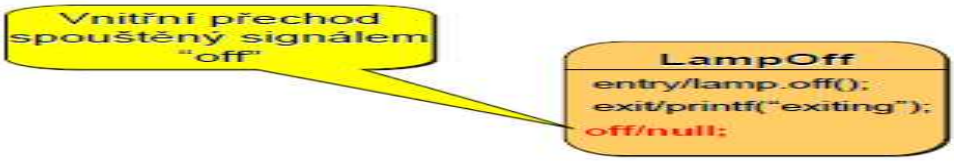


Vstupní a výstupní akce stavu

Pořadí provádění akcí

Vnitřní přechody

- Nedochází k přechodu do jiného stavu



„Do“ Aktivity

- Spouští paralelní vlákno prováděné dokuk:
  - Akce neskončí
  - Dojde k přechodu do jiného stavu (signálem)

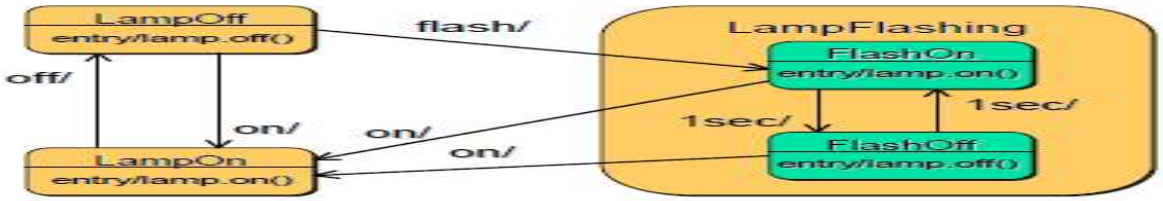
Podmíněné přechody

- Stimul + podmínka

Podmíněné větvení

Hierarchické stavové automaty

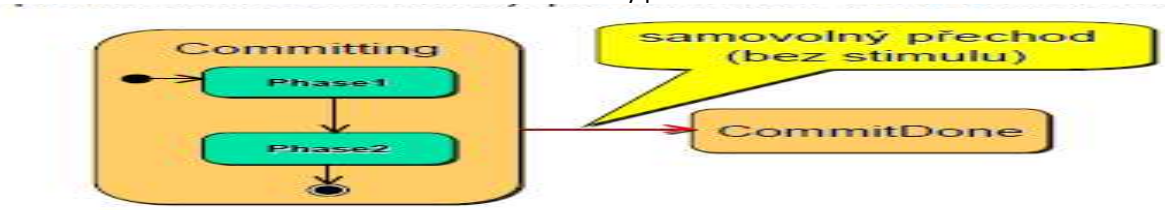
- Stav představuje automat



Skupinové přechody

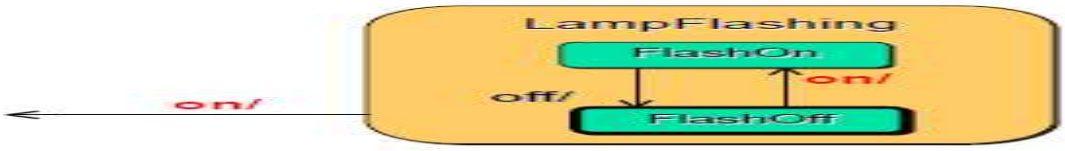
Samovolný přechod

- Nemá stimul
- Proveden automaticky po skončení vnitřního stavu

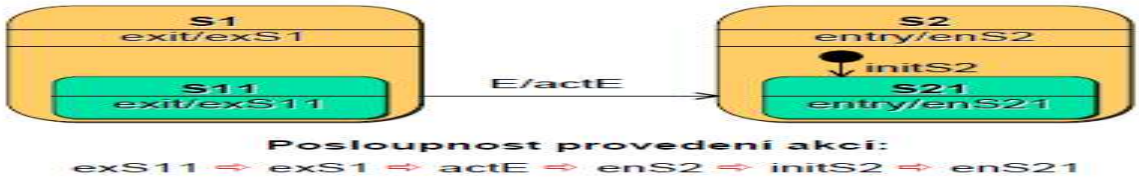


Spouštěcí pravidla

- Více přechodů může mít stejný signál
  - Vnitřní mají přednost



Pořadí akcí

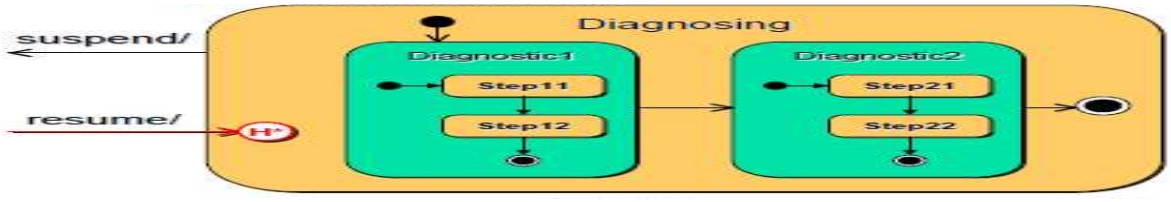


Posloupnost provedení akcí:

exS11 ⇨ exS1 ⇨ actE ⇨ enS2 ⇨ initS2 ⇨ enS21

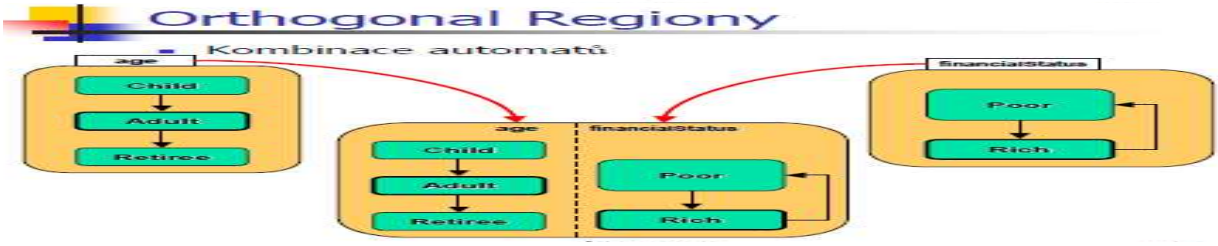
Historie stavů

- Chceme se vrátit do místa, odkud jsem opustili složený stav
  - Hluboká a mělká historie



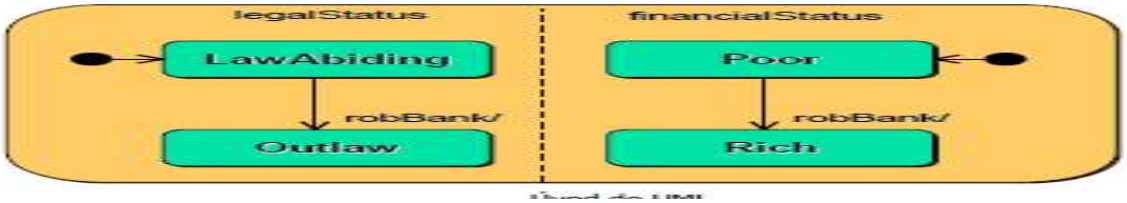
Orthogonalita

- Entita je současně ve více nezávislých stavech



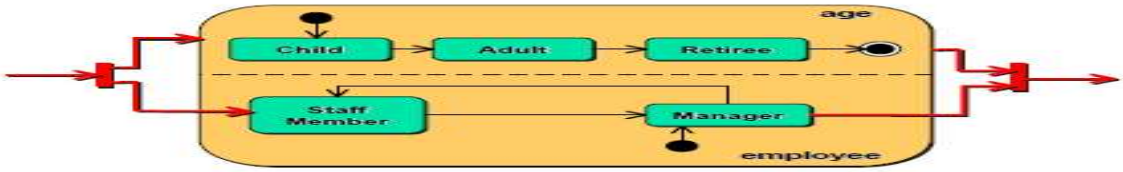
Sémantika OR

- Automaty v regionech přijímají oba vnější podněty a běží paralelně

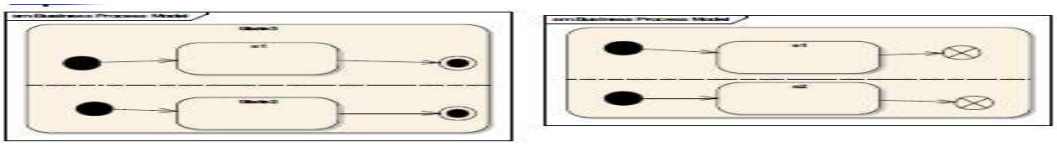


Sémantika

- Čeká se na doběhnutí obou automatů

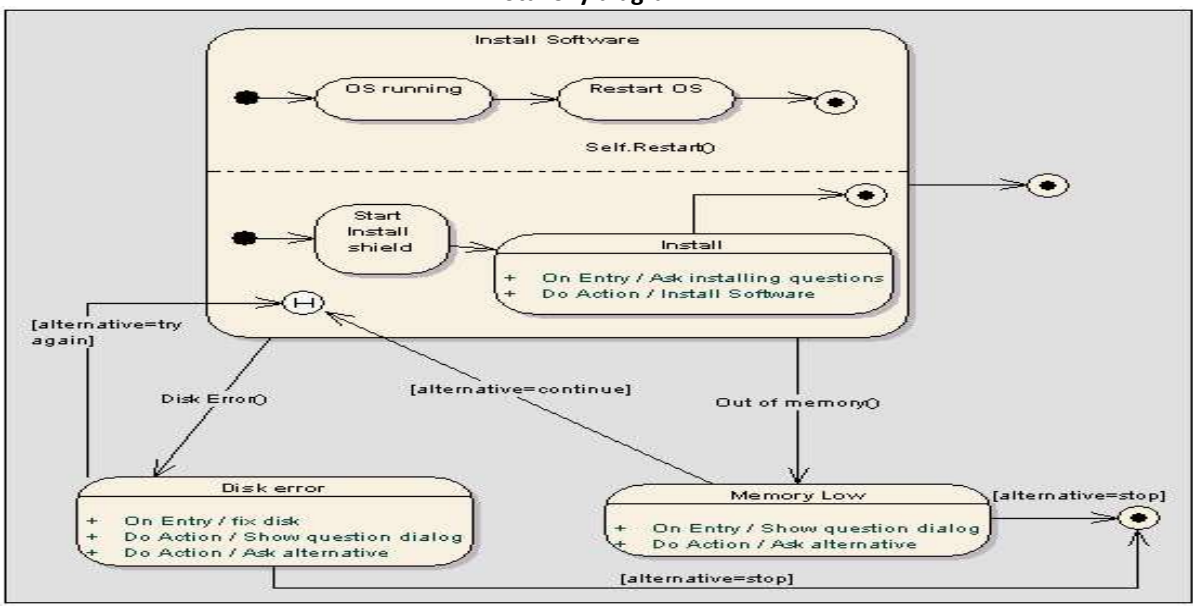


Podmínka na ukončení

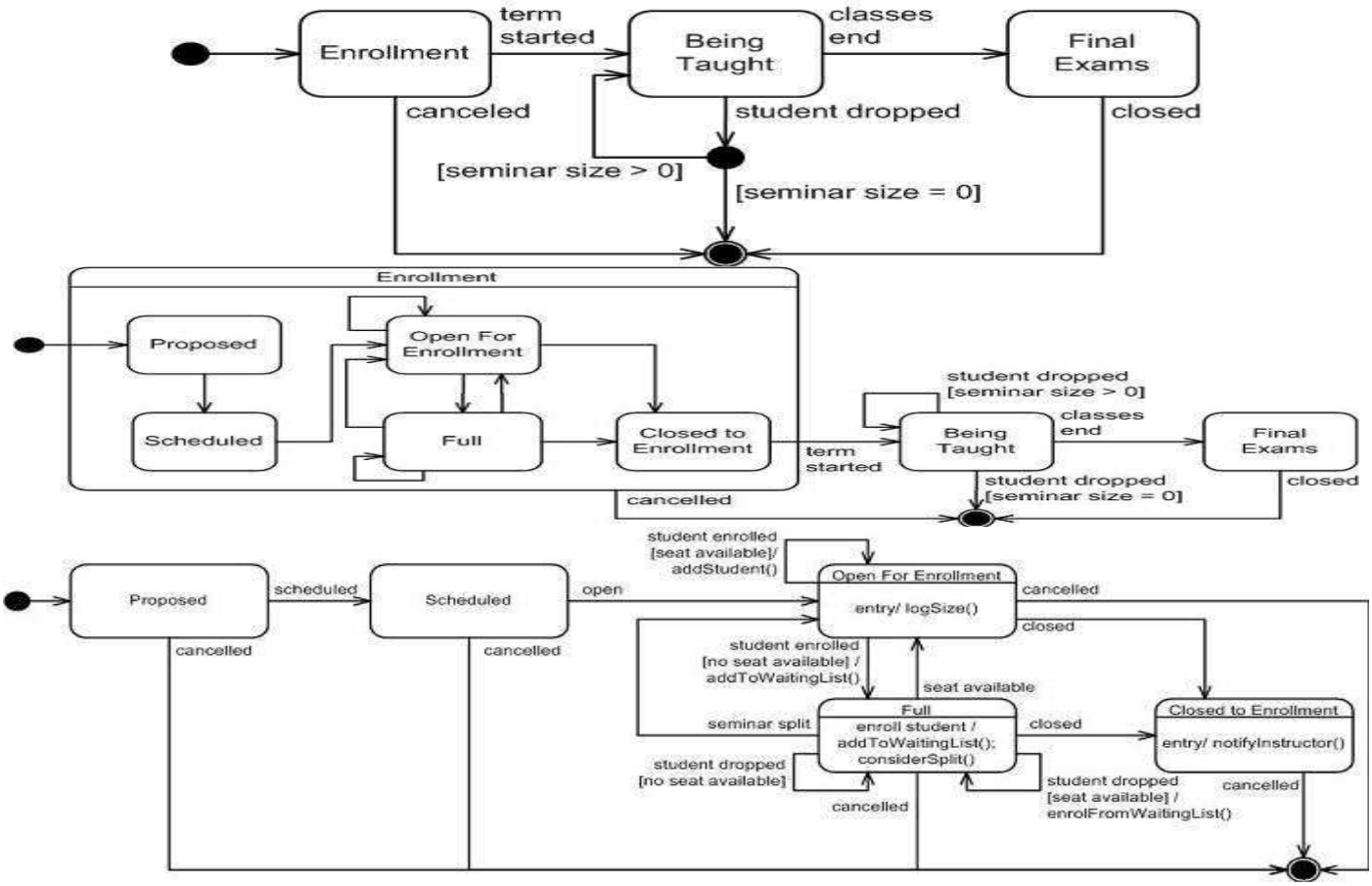


- AND - OR

Stavový diagram

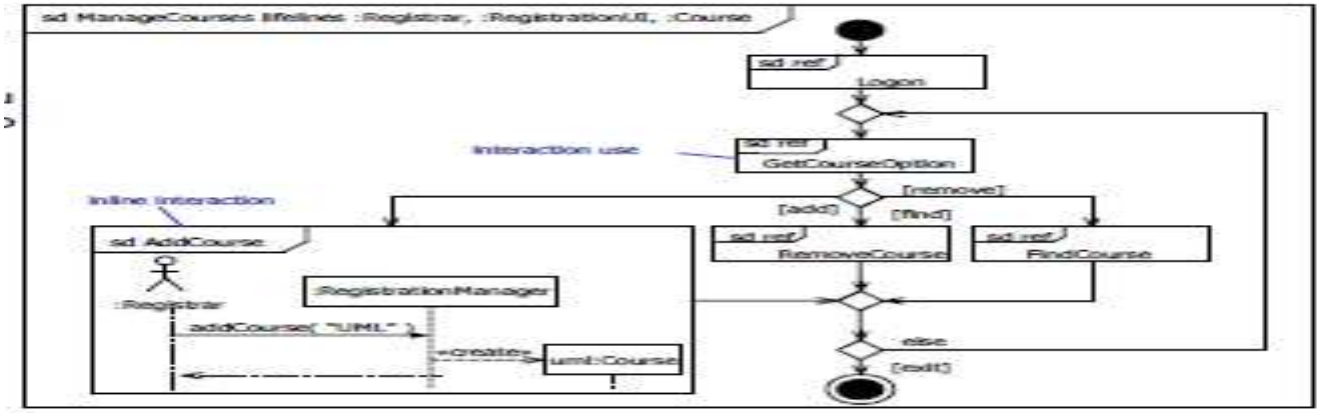


Příklad



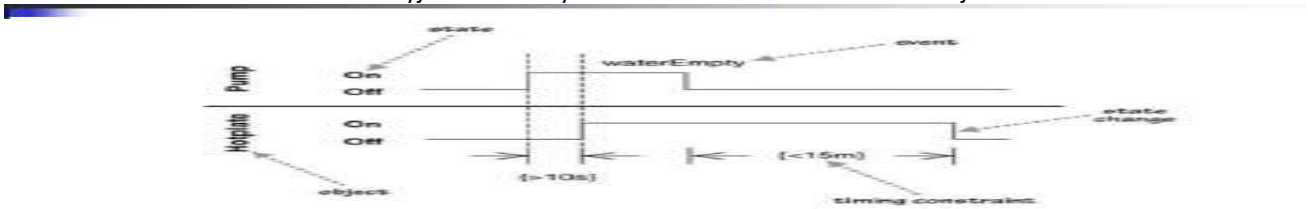
Přehledové diagramy interakce

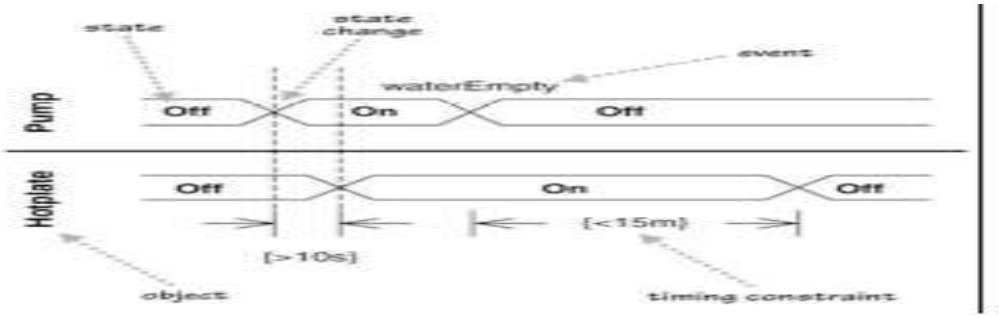
- Zobrazuje tok mezi interakcemi
- Kombinace diagramu aktivit a sekvenčního diagramu



Časové diagramy

- Vyjádření časových závislostí mezi změnami stavů objektů




state
state change
event
waterEmpty
Pump
Off
On
Off
Hotplate
Off
On
Off
{<15m}
{>10s}
object
timing constraint