UML

Historie:

- Programovací jazyky

Simula	1962	třída,objekt, polymorfismus, dědění	Nejspíš první OOP jazyk, odvozený z Algolu. 1967 – Systémy hromadné obsluhy. Neuplatnil se mimo akademické prostředí.
SmallTalk	1972	virtuální stroj	OOP Jazyk, vyvinut firmou Xerox. Inspirován právě Simulou. Komunita kolem SmallTalku vyvinula extrémní programování a také například návrhové vzory.
C++	1983	vyjímky	Používán od roku 1980, název vymyslel Rick Mascitti. První oficiální forma jazyka přijata až 1998
Eifel	1986	garbage collection	

- Objektové modelovací jazyky a metodiky

Raumbach	1991	OMT
Jacobson	1992	Objectory->později firma Rational->UML
Booch	1994	OOSE (Object Oriented Software Engineering

1994 Zahájení vývoje UML ve firmě Rational Software (Raumbach, Booch), která dneska patří pod IBM, kdy oba pánové začali spojovat své metodiky, tedy OMT (Object Modeling Technique) a Booch.
1995 Notace pro Unified Method 0.8,**0.9** + Jacobson vstupuje do Rational se svým OOSE -> model RUP
1996 UML 1.0
1997 UML 1.1 - standard OMG
1999 UML 1.3
2001 UML 1.4 - komponenty
2003 UML 1.5 - diagramy aktivit a sémantika akcí
2005 UML 2.0 - nové diagramy
2006 UML 2.1 - další snahy o upřesnění software

1.Co je to vlastně UML?

---

1)Co je to UML?

Jazyk UML (Unified Modeling Language) je unifikovaný modelovacví jazyk pro vizuální modelování systémů a byl navržen tak, aby spojil nejlepší existující postupy modelovacích technik a softwarového inženýrství. Je navržen tak, aby jej mohli implementovat všechny CASE nástroje. Jazyk se zrodil v roce 1994 a od roku 1997 se stal profesionálně uznávaným standardem. Grafický jazyk pro vizualizaci

---

2)Co je to unifikace?

Nabízí modelování po celou dobu životnosti softwarového projektu a na jeho požadavky pro analýzu počínaje a implementací konče. Je to jazyk, který umožňuje modelovat jak systémy zasazené do reálného času tak i podpůrné systémy rozhodování a je nezávislý na jekémkoliv programovacím jazyce a na jakékoliv platformě. Má však vynikající podporu OOP jazyků jako je C# nebo Java. Kromě Unified Proces můžeme v jazyce UML modelovat i jiné osnovy tvorby softwarového inženýrství a UML se o konzistenci snaží dále prostřednictvím množiny interních pojmů

Sjednocuje předchozí modely a podporuje celý vývojový cyklus
Volnost v použití metodiky
Založeno na zkušenostech a potřebách uživatelů

---

3)Co není UML?

Metodikou = nepředepisuje, jak postupovat při vývoji software! Při každém modelování musíme vymyslet UML úplně od začátku. Neexistují hotová řešení.

---

4)Co obsahují modely UML?

Stavební bloky - jsou to základní prvky modelu: Relace a Diagramy
Společné mechanismy - obecné mzpůsoby, jimiž se v jazyku UML dosahuje specifických cílů

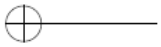
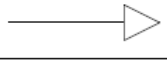
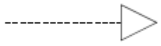
Architekturu
Tento fakt zdůrazňuje uspořádání samotného jazyka UML = který se tak stává taktéž navrhovaným a sestavovaným systémem.

Dva základní aspekty modelu UML jsou
a)Statická struktura – popisuje, jaké typy objektů jsou pro modelování daného systému důležité a jednak jak spolu tyto objekty souvisí
b)Dynamické chování – popisuje životní cyklus zmiňovaných objektů a jejich vzájemné spolupráce s cílem dosáhnout požadované funkce navrhovaného systému.

---

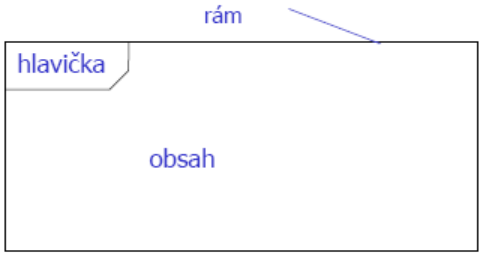
5)Z čeho se skláká UML?
a)Předmětů - Strukturní abstrakce samotné prvky modelu,jsou v UML vyjádřeny podstatnými jmény a chování v modelu UML je vyjádřeno slovesem. Balíček, tedy jediná abstrakce seskupování se používá pro seskupení významově souvisejících předmětů. Poznámky nám v modelu mohou sloužit k podobnému účelu, jako zvýrazňovač

b)Relací - Propojují jednotlivé předměty. Vztahy/Relationships

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.

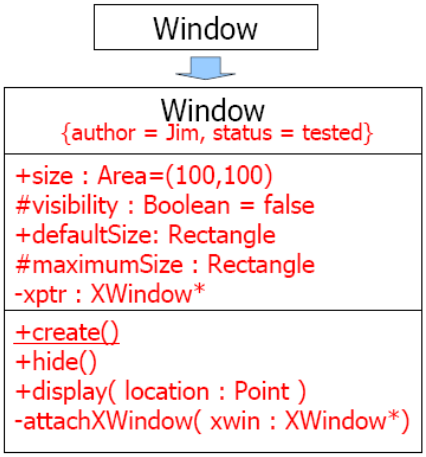
c)Diagramů - Znázorňují zajímavé pohledy na model. Vizualizace toho, co a jak bude systém dělat. Diagram NENÍ model!!! Předměty a relace lze z diagramu odstranit. V UML existuje 13 základních diagramů: Diagram struktury a chování, diagram tříd, diagram komponent, diagram nasazení, diagram aktivit a jiné.

Syntaxe diagramu:



---

6)Jaké mechanizmy obsahuje UML?
a)Specifikace - Testové popisovací funkce a sémantiky jednotlivých prvků použitých v modelu (jádro modelu) - jádrem a podstatou UML modelu. Nabízejí sémantický základ modelu.
b)Ornamenty - Informace o prvku modelu, aby ukázaly jeho význam.



- jednotlivé prvky modelu zdobíme v diagramech UML proto, abychom zdůraznili důležité detaily

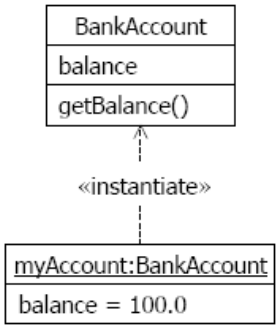
c)Podskupiny-

Klasifikátory - Abstraktní vyjádření předmětu (bankovní účet)

Instance – konkrétní, specifický výskyt typu předmětu (můj bankovní účet), opakem klasifikátoru.

Rozhraní - dohoda, která specifikuje chování předmětu, tlačítka na čelním panelu rekordéru

Implementace - specifikuje podrobnosti chování tohoto předmětu, mechanismus uvnitř rekordéru.



Mechanizmy rozšíření

Protože návrh naprosto univerzálního jazyka je prozatím nemožný, zahrnuli tvůrci jazyka UML tato 3 rozšíření, které umožňují jeho rozšiřitelnost.

a)Omezení/Constraints - umožňují přidávat nová pravidla k prvkům modelu

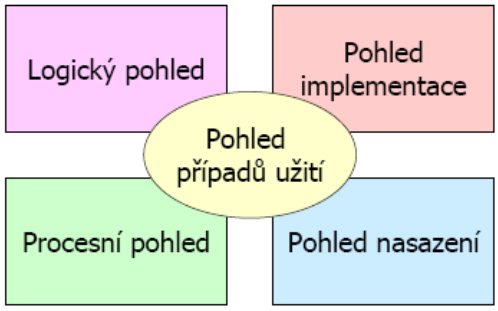
b)Stereotypy - zavádějí nové prvky modelu založené na prvcích existujících

c)Označené hodnoty - doplnění modelů o nové vlastnosti, kdy označená hodnota je klíčové slovo s přidruženou hodnotou. Dodatečná informace přičleněná k modelu, např: [autor = Jim]

---

7)Co je to architektura?

Architektura je vlastně organizační struktura systému včetně jeho rozkladu na součásti. Dle standardu IEEE je architektura definována jako nejvyšší úroveň koncepce systému v jeho vlastním prostředí. Pohled „4+1“ Všechny 4 pohledy jsou integrovány do pátého pohledu a to do případů užití.



a)Logický pohled – Koncový uživatel a Tester. Funkce systému a slovník

b)Pohled procesů –Analytik a Integrátor systému, výkon systému, škálovatelnost a propustnost

c)Pohled implementace – Programátor a Tester. montáž procesů a správa konfigurace

d)Pohle nasazení - topologie systému, distribuce, doručení a instalace

2.Co je to Unified Process (UP)?

---

8)Co je to UP?
Unified Proces. Proces tvorby softwarového vybavení převádí uživatelské požadavky na software, přičemž specifikuje KDO CO dělá. Tato metodika byla vyvinuta 1967. Komerčním UM se stala metodika RUP a je zpětně kompatibilní.

---

9)Je možné aplikovat metodiku UP na každý projekt?
Pro každý projekt je třeba vybudovat zcela odlišnou instanci UP.
Vize a požadavky -> Proces softwarového inženýrství -> Software

---

10)Čím je charakteristická metodika tvorby softwarového vybavení?
a)rizika
b)architektura
c)iterace

je řízena rizikem a případy užití = požadavky, soustředuje se na architekturu. Je iterativní a přírustková (inkrementační). To jsou tři základní axiomy UP. Rizika projektu řeší manažer projektu, a nebo tvůrce projektu.
Iterace = rozklad projektu na menší dílčí části.

---

11)Co se myslí tím, že je software vytvářeno v iteracích?
Každá iterace je jako samostatný miniprojekt, který dodává jednotlivé součásti vznikajícího systému. Iterace jsou skládány jedna na druhou, čímž vytvářejí konečnou podobu systému.

---

12)Co může obsahovat každá iterace?
Iterace obsahuje požadavky, které slouží k tomu, aby popsaly, co má systém umět. Dále obsahuje analýzu, která upřesňuje a strukturuje požadavny na daný systém. Obsahuje návrh, který převádí požadavky na architekturu systému, dále pak implementaci na níž je ověřena funkčnost softwaru, který je tímto vytvořen. Na závěr obsahuje iterace testování, během kterého je ověřena většina vytvořených a poskytovaných funkcí.

---

13)Z jakých fází se skládá metodika UP?
Metodika UP se skládá ze zahájení, fáze, během níž projekt startuje,
dále z fáze rozpracování, tedy fáze, kdy vzniká architektura systému.
Fáze konstrukce, tedy fáze, během níž vzniká samotný software a milníkem je funkční varianta programu
a na závěr je tu ještě fáze zavedení, tedy fáze, kdy je software zaveden do uživatelského prostředí a milníkem je nasazení produktu.

3.Požadavky

---

14) V jakých fázích se vytvářejí požadavky?
ve fázi zahájení (sběr požadavků funkčních a nefunkčních) a rozpracování projektu v metodice UP

---

15) Jaký je metamodel požadavků?
Funkční a nefunkční požadavky
Aktéři a případy užití

---

16) Jaké aktivity jsou součástí pracovního postupu tvorby požadavků?
Vyhledání aktérů a případů užití
Zobrazení detialů případů užití - scénáře a alternativní scénáře
Struktura modelu případů užití

---

17) Jak lze rozšířit pracovní postup tvorby požadavků?
Tvůrcem požadavků je aktér
aktivita: vyhledání funkčních požadavků
aktivita: vyhledání nefunkčních požadavků
aktivita: stanovení priorit jednotlivých požadavků
aktivita: sledování požadavků až k případům užití

---

18) Co je častou příčinou nezdaru softwarových projektů?
Většina projektů končí nezdarem v důsledku problémů, které vznikly v procesu inženýrství požadavků, tedy že se špatně stanovily všechny požadavky na systém a nebo nebyly formulovány přesně a došlo k jejich chybné implementaci nebo nepochopení významů některých problémových pojmů

---

19) Jaké jsou základní typy požadavků?
Funkční a nefunkční
Funkční popisují požadavky na požadovanou službu systému a nefunkční definují vlastnosti systému a omezení, za nichž musí pracovat

Příklady funkčních požadavků:
Bankomat bude ověřovat validitu vložené platební karty
Bankomat vydá na každou kartu maximálně 10.000 Kč během 24hodin

A příklad nefunkčních požadavků:
Řídicí systém bude napsán v jazyce C++
Řídicí systém ověří validitu PIN kódu nejdéle do tří sekund od zadání

---

20) Jak se správně formulují požadavky?
Správně se požadavky formulují jednoduchým strukturovaným jazykem (angličtinou, čechoangličtinou) s užitím klíčového slova "bude umět"nebo anglicky "shall" takže schéma pak vypadá takto
<id><systém> bude <funkce>

---

21) Jakou podobu mohou mít systémové požadavky?
Specifikace obsahuje jak funkční tak nefunkční požadavky, může mít pak podobu dokumentu nebo databáze v nástroji pro správu požadavků

---

22) Jak lze uspořádat požadavky?
Požadavky lze uspořádat do taxonomie, do hierarchie typů požadavků, která kategorizuje požadavky.

---

23) Jaké atributy lze přidat k požadavkům?
Atributem je myšleno dodatečná informace, metadata, která jsou k požadavku přidružena. Například: nezbytné, možné, eventuální, chceme mít. nebo atributy z modelu RUP a to status,význam, snahu, riziko, stabilitu, cílovou verzi

---

24) Co to jsou univerzální kvantifikátory?
Mohou naznačovat hranice duševní mapy ve vidění okolního světa. Platnost takovýchto klíčových kvantifikátorů je třeba si ověřit. Kvantifikátory se vyznačují touto významovou stavbou: všichni,každý,vždy,nikdy,nikdo a žádný. Kdykoliv se setkáme s univerzálním kvantifikátorem máme co dočinění s vyřazením,zkreslením nebo zobecněním a platnost takovýchto vět je třeba si ověřit.

Například:
Každý, kdo si chce půjčit knihu v knihovně, musí mít průkazku. – klasický příklad zobecnění
Pochybnost: Existují případy, kdy by si někdo bez průkazky půjčil knihu?
Odpověď: Třeba uživatelé z jiných knihoven nemusí mít průkazky nebo mají speciální typ s jinými podmínkami

---

25) Jaké jsou techniky hledání požadavků?
Pohovory, dotazníky (nenahradí osobné pohovory), dílny požadavků
musíme zjistit, kdo jsou přímí uživatelé systému a jaké jsou další zainteresované osoby (správci, údržba) další systémy s nimiž bude náš vyvýjený systém v kontaktu,hardwarová zařízení v iteraci a právní a regulační omezení vztahovaná na náš projekt a taky jaké jsou obchodní cíle. Všechny nápady jsou přijímany za dobré!

4.Modelování případů užití

---

26) Jakého pracovního postupu jsou součástí případy užití?
Pracovního procesu Požadavky.

---

27) V jakých fázích se provádějí práce na případu užití?
V prvotních cyklech tedy - Zahájení a rozpracování projektu

---

28) Jaké jsou klíčové aktivity při tvorbě případů užití?
Nalezení aktérů a případů užití
Detial případů užití, tedy jeho čistý průchod a případně alternativní scénáře případů užití, pokud jsou třeba.

---

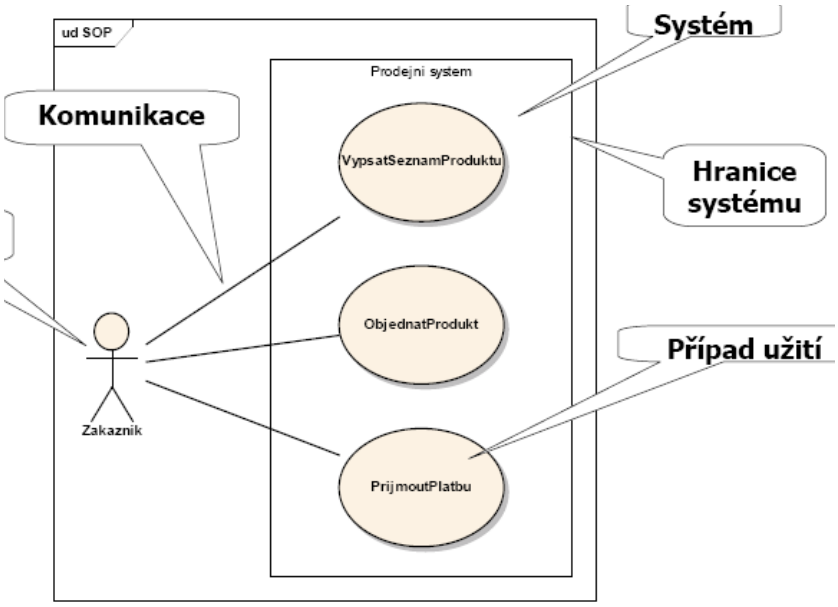
29) Jaké jsou etapy modelování připadů užití?
Nalezení subjektu/problému?
Nalezení aktérů
Nalezení případů užití

---

30) Co jsou to aktéři a jak se hledají?
Role přidělené entitám, které přímo komunikují se systémem. Kdo nebo co používá systém a aktérem je velmi často také čas.

---

31) Co jsou případy užití?
Funkce, které systém vykonává jménem jednotlivých aktérů nebo v jejich prospěch. Nacházíme je prostřednictvím úvah na téma jaké funkcenáš systém uživatelům nabízí a jaké případy užití vždy iniciuje aktér a případy užití jsou vždy psány z pohledu aktérů



---

32) Co znázorňují případy užití?
Subjekt, aktéry, případy užití a interakce

---

33) Co je a k čemu slouží slovníček pojmů?
Project Glossary poskytuje definice klíčových slovních obchodních termínů a obsahuje definice synonym a homonym

---

34) Co obsahuje specifikace případu užití?
Název, jedinečný kvantifikátor, krátký popis, cíl případů užití a aktéry: hlavní a vedlejší.
Ti hlavní případ užití inicializují a ti vedlejší jsou v iterakci s případem užití po jeho spuštění.
Vstupní podmínku
Hlavní scénář - kroky případu užití
Výstupní podmínky a alternativní scénáře

---

35) Jak lze omezit počet případů užití?
Počtem rozvětvení jednotlivých scénářů a u větví se použije klíčové slovo KDYŽ (If)
Pokud chceme zachytit větvení, ke kterému může dojít kdykoliv použijeme konstrukci: Alternative Flow - alt.scénář.

---

36) Co je to komplexní případ užití?
Hlavní scénář, který předpokládá, že během užití nedojde k ničemu nepředvídatelnému a jeden nebo více alternativ

---

37) K čemu slouží vedlejší scénáře?
Hledání alternativ, chyb a přerušení

---

38) Kdy se případy užití rozkládají?
Rozkládáme pouze tehdy, pokud tím obohatíme náš model

---

39) Co je to matice sledovatelnosti požadavků?
Požadavky v modelu požadavků mohou být k případům užití sledovány pomocí matice sledovatelnosti požadavků
Ověříme si tak snadno, zda jsme namodelovali všechny požadavky, které jsme měli zadány a zda nikde nic nechybí.

---

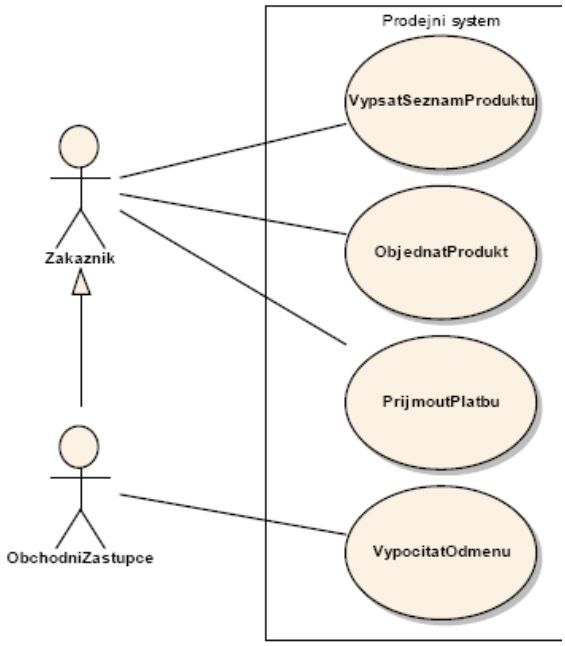
40) Kdy je vhodné modelování případů užití?
Kde převládají funkční požadavky.
Když se vyskytuje mnoho aktérů systému.
Když je v modelu obsaženo mnoho rozhraní k dalším systémům.

a kdy je naopakmodelování nevhodné, když:
Převládají nefunkční požadavky
Je tam málo aktérů a málo obslužných rozhraní

---

41) Co umožňuje zobecnění akterérů?
Generalizace
Umožňuje vyčlenění aktérů a jejich chování, které je společné dvěma a více aktérům do jednoho rodičovského aktéra.
Rodičovský aktér je obecnější a potomci jsou specializováni
Zděděného aktéra můžeme dosadit tam, kde očekáváme přdka
Rodičovský aktér je víceméně abstraktní - specifikuje abstraktní roli
Zobecění případů užití může případy užití zjednodušit.

Ukázka zobecnění aktérů



---

42) Co umožňuje vyčlenit společné funkce případů užití?
Vyčlenit rodičovské případy užití.
Odvozené případy užití dědí všechny vlastnosti a funkce od svých předků. A mohou být doplněny o nové funkce a vlastnosti
V odvozených případech užití se používá jednotná konvence:
Zděděný prvek beze změny 3. (3.)
Zděděný a přečíslovaný prvek 6.1 (6.1)
Zděděný a překrytý prvek 1. (01.)
Zděděný, překrytý a přečíslovaný 5.2 (05.1)
Přidaný prvek 6.3

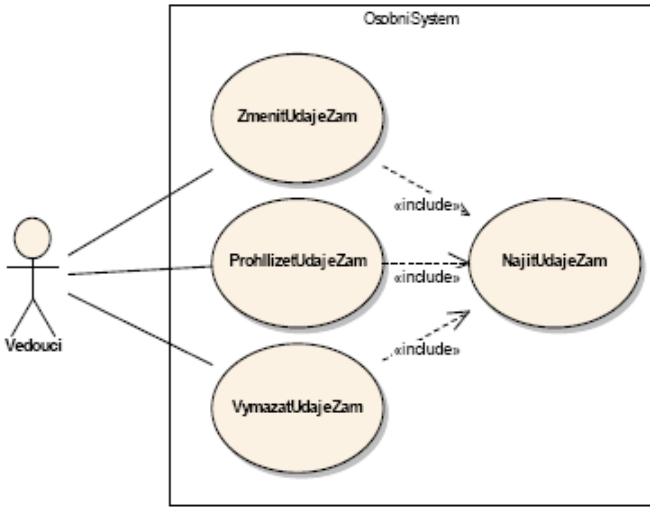
Dobrým zvykem je abstraknost rodičovských případů užití.

---

43) Co umožňuje relace "include"?

Opakující kroky vyčlení do samostatného případu užití a vytvoří z něj případně bázový případ užití

Ukázka <<include>>

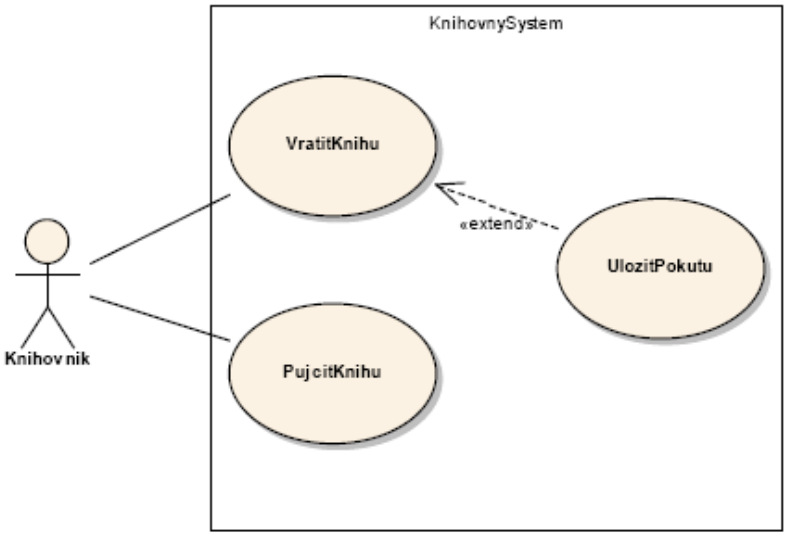


---

44) Co umožňuje realce "extend"?

Přidává do existujicího případů užití nové chování. Je to místo rozšíření

Ukázka <<extend>>



---

45) Jaké používáme pokročilé funkce při modelování případů užití?

Zobecnění aktérů - pokud nám zjednodušší model

Zobecnění případů užití - doporučeno nepoužívat, pouze pokud pracujeme s abstraktními předky

Relace <include> - Pozor na nadměrné užití

Relace <extend> - doporučeno nepoužívat. Pokud jí chceme použím, máme zajistit, aby sémantika byla všem jasná.

---

46) Jaké znáte rady pro psaní případů užití?

Být stručný a držet se jednoduchosti. Soustředění se na otázku CO nikoliv JAK

a vyhýbání se funkční dekompozici - pokud uvažujeme o modelu špatným způsobem. Pokud jsme například zběhlejší v programování a o projektu přemýšlíme jako programátoři nikoliv jako analytici z hlediska OOP

Hierarchie případů užití by neměly být hlubší než 2 úrovně!!

6.Analýza

---

47) Jaké jsou artefakty analýzy?

a)Analytické třídy, které tvoří klíčové pojmy v obchodní doméně

b)Relace příparů užití - ukazuje vzájemnou komunikaci analytických tříd a cílem realizace chování systému specifikovaného Use Case Diagramu.

---

48) Jaké aktivity zahrnuje analýza podle metodiky UP?
Architektonická analýza
Analýza případů užití
Analýza tříd
Analýza balíčku - třídy (obdélníky) a use casy (elipsy)

Vývoj software: Požadavky -> Analýza -> Návrh -> Implementace -> Testování

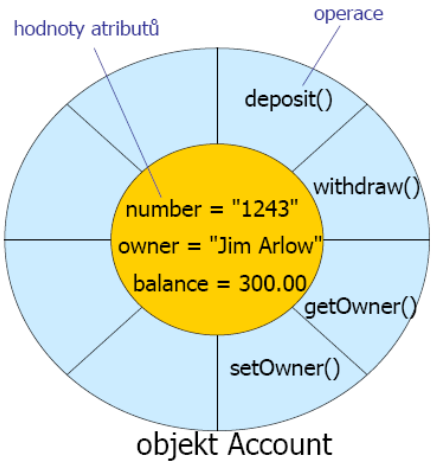
---

49) Jaké jsou doporučení pro vytváření analytického modelu?
Analytický model středně velkého systému obsahuje přibližně 50-100 tříd
Do modelu by měly být zahrnuty pouze třídy, které modelují slovníček pojmů problémové domény
Nestarat se o implementaci
Minimalizace vazeb a asociací mezi třídami
Používat dědičnost, kde to dovoluje přirozená hierarchie abstrakcí
Udržovat model co nejjednodušší

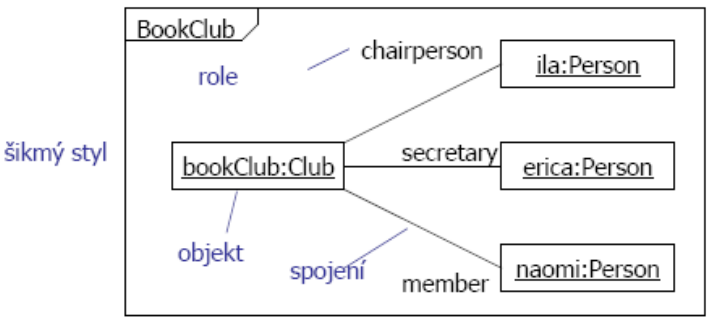
7.Třídy a objekty

---

50) Co jsou to objekty?
Soudružné jednotky které snoubí data s funkčností



Diagramy objektů:



to samé může být zapsáno i jiným stylem čar.

---

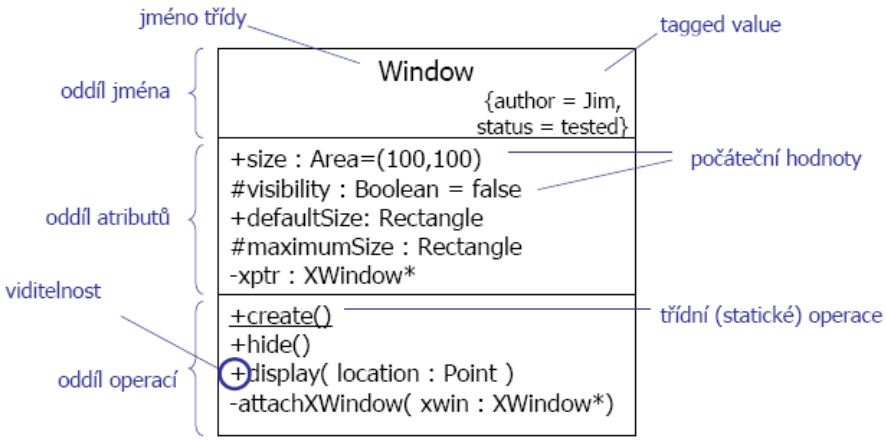
51) Jaký je rozdíl mezi operacemi a metodami?
Operace = abstraktní specifikace pro funkce objektu vytvořených během analýzy
Metody = jsou kontrétní specifikace funkcí objetu vytvořených v etapě návrhu

---

52) Co je to instance třídy?

Objekt, třída definuje společné vlastnosti sdílené všemy objekty dané třídy

Notace třídy:



---

53) Jaké jsou vlastnosti objektů?

a)Identita - jedinečná identifikace existence objektu

b)Stav - smysluplná množina atributů objektu v určitém čase

- Objekt (BankovniUcet...zustatek < 0 --> stav:PrecerpanyUcet)

c)Chování - je vyjádřením služeb objektu poskytovaných dalším objektům

---

54) Co je vzájemná interakce jednotlivých objektů?

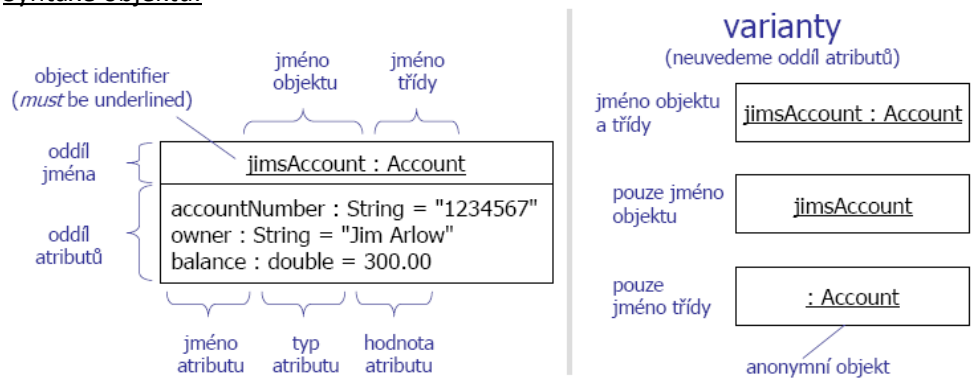
Vzájemná interakce jednotlivých objektů systému generuje: Výsledné chování celého systému

Interakce zahrnuje objekty zasílající a přijímající zprávy. Po přijetí zprávy je automaticky volána odpovídající metoda, která může způsobit přechod stavu z jednoho do druhého.

---

55) Jaká je notace objektů v jazyce UML?

Syntaxe objektů:



Každý symbol v jazyce UML má dva oddíly

Horní oddíl obsahuje název objektu a nepovinně název třídy.

Všechna slova musí být podtržena. Názvy objektů jsou tvořeny velbloudím písmem a první písmenko je velké.

- názvy objektů začínají malým písmenem, a je od třídy oddělen dvojtečkou
- názvy tříd velkým písmenem

Spodní oddíl obsahuje názvy atributů a jejich hodnoty přiřazené k názvům znakem rovnítka

Typy atributů se v diagremech vynechávají

Psáno vše velbloudím písmem a malým písmenem na začátku

---

56) Co definuje třída určité množiny objektů?

Charakteristické vlastnosti: Atributy, operace, metody, chování a relace

Proces, v němž jako šablonu pro tvorbu nového objektu používáme třídu

Každý objekt je instancí přesně jedné třídy.

Různé objekty třídy se vyznačují stejnou množinou atributů, které mohou ovšem nést jiné hodnoty, díky tomu se objekty mohou chovat různě. Jinak se bude chovat objekt bankovní účet s přečerpaným stavem a jinak účet, na kterém je mnoho peněz.

Instanční relaci mezi třídou a jejím objektem lze vyjádřit stereotypem: <<instantiate>>

---

57) Jaká je notace tříd v jazyce UML?

oddíl názvu
- stereotyp
- název třídy - neměl by obsahovat zkratky
- označené hodnoty

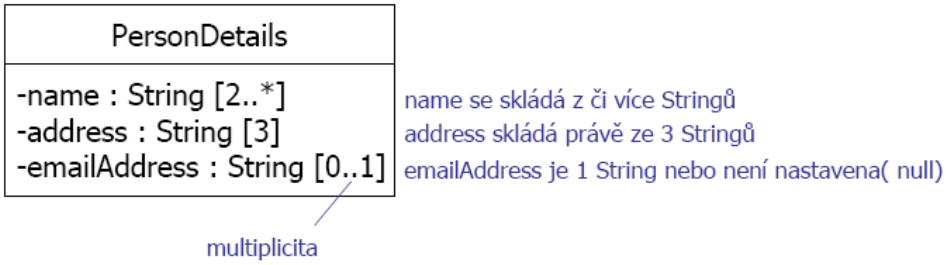
oddíl atributů
- inicializační hodnoty atributů, násobnost, například při používání polí

oddíl operací
- vyditelnost operací. plusko nebo mínusko
- operace třídy jsou podtržené
- název
- seznam argumentů, implicitní hodnoty argumentl, směr - in, out, inout, return

vlastnost

signatura

Ukázka násobnosti:



---

58) Co je rozsahem platnosti instance a třídy?

Atributy a operace se vztahují pouze ke specifickým objektům.
- Operace instance mohou používat další atributy nebo operace instance
- Operace instance mohou používat VSECHNY atributy nebo operace mající platnost třídy

Atributy a operace se vztahují na celou třídu objektů
- Operace třídy mohou používat další atributy nebo operace třídy

8.Hledáme analytické třídy

---

59) Co vyjadřuje analytické třídy?

Vyjadřují velmi přesně definované zobecnění problémové domény. Problémovou doménou je doména, ze které vzešel požadavek na vznik nového systému. Analytické třídy by měli být mapovány na pojem užívaný v reálném světě. Obchodní pojmy používané v analytickém modelu je dobré si ujasnit a vysvětlit jejich význam

---

60) Co je výstupem aktivity analýza případů užití?

Analytické třídy a realizace případů užití

---

61) Co obsahuje analytický model?

Obsahuje pouze analytické třídy, tento model nesmí obsahovat žádné třídy vzniklé v návrhových úvahách

---

62) Co obsahují analytické třídy?

Množinu hlavních kandidátů - atributů
Množinu hlavních operací

---

63) Jak se pozná dobrá analytická třída?

V názvu zrdcadlý svůj účel, zobecnění, které modeluje jeden specifický prvek v problémové doméně. Definuje zřetelně vlastnosti problémové domény. Vlastní malou množinu odpovědností.
- dohoda nebo závazkem třídy vůči klientům
- soudržná množina operací a každý třída by jich měla mít 3-5!!!

Malý počet vazeb do okolních tříd => třída je soběstačná

---

64) Jak se pozná špatná analytická třída?

Funktoid - třída s jedinou operací
Všemohoucí třída - děvečka pro všechno
Obsahuje široce rozvětvený strom dědičnosti
Málo soudružná

---

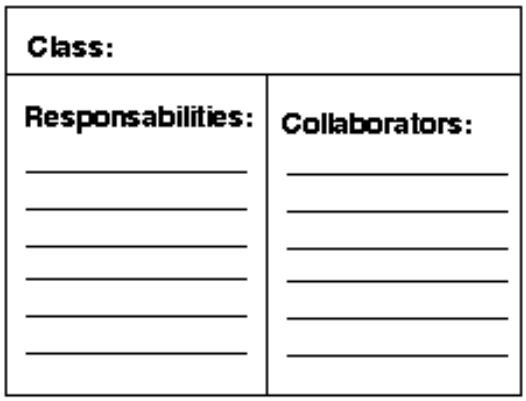
65) Popište metodu "Analýza podstatnými jmény a slovesy"

Hledáme všechna podstatná jména a spojení podstatných jmen -> kandidáti na třídy a atributy
Hledáme slovesa a slovesné fráze -> kandidáti na odpovědnosti a operace
Následuje analýza sběru podst.jmen a sloves

---

66) Popište metodu "CRC"

Metoda štítků
Je výkonná, spontánní diskuze o hledání nápadů, všechny nápady.Každý lísteček je rozdělen na 3 oddíly:
a)oddíl třídy
b)oddíl odpovědností
c)oddíl spolupracovníků
Spontánně se diskutuje o náhlých inspiracích, pojmenování předmětů se kterými se v obchodní doméně pracuje a uvedení odpovědností
na lísteček k dané doméně atd....



---

67) Popište metodu "Sterotypy RUP"

<<boundary>> - Třída je prostředníkem mezi systémem a jeho prostředím
<<control>> - Třída zapouzdřuje chování příznačné pro daný typ případu užití
<<entity>> - Třída se pouzívá k modelování perzistentních informací o něčem, poskyuje informace

---

68) Jaké zdroje můžeme použít při hledání analytických tříd?

fyzické objekty, kancelářské pomůcky, rozhraní k vnějšímu světu a koncepční entity

9.Relace

---

69) Co je to relace?

Sémantická vazba mezi předměty a abstrakcemi

---

70) Co je to spojení?

Vazba mezi objekty.
Ke spojení dochází vždy, pokud jeden objekt obsahuje odkaz na další objekt
Objekty uskutečňují chování modelovaného systému díky vzájemné spolupráci
- předávání zpráv mezi nimi
- objekt po přijetí zprávy příslušnou metodou.

---

71) Co ukazují objektové diagramy?

Ukazují subjekty a jejich významná spojení v určitém okamžiku
Jsou to snímky běžícího objektově orientovaného systému
Mohou vůči sobě ty objekty hrát role a tyto role definují sémantiku jeho úlohy při spolupráci

---

72) Co je to asociace a jaké je syntaxe v UML?

Sémantická vazba mezi třídami

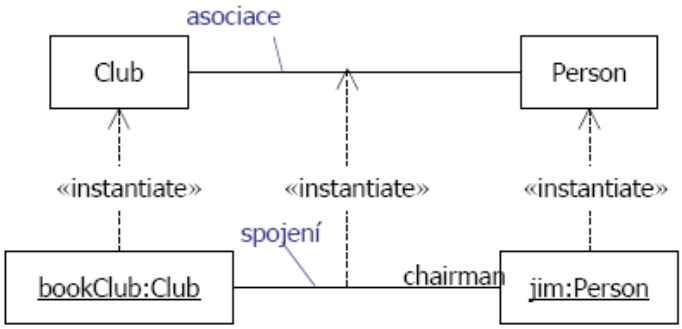
„Jeli mezi dvěma objekty spojení, musí existovat rovněž asociace mezi třídami těchto objektů“.!!!

Spojení jsou instance asociací. Spojení je u objektů a asociace je u tříd.

Asociace mohou mít následující syntaxe:

- název asociace
  - se směrem asociace
  - se slovesem nebo slovesnou frází
  - s černou šipkou naznačující směr asociace
  - vždy začínat malým písmenem
- názvy rolí na jednom nebo na obou koncích
- průchodnost
- násobnost na jednom nebo na obou koncích

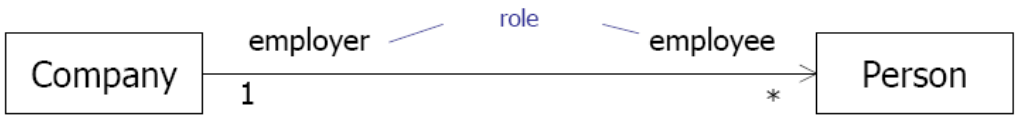
Příklad asociace:



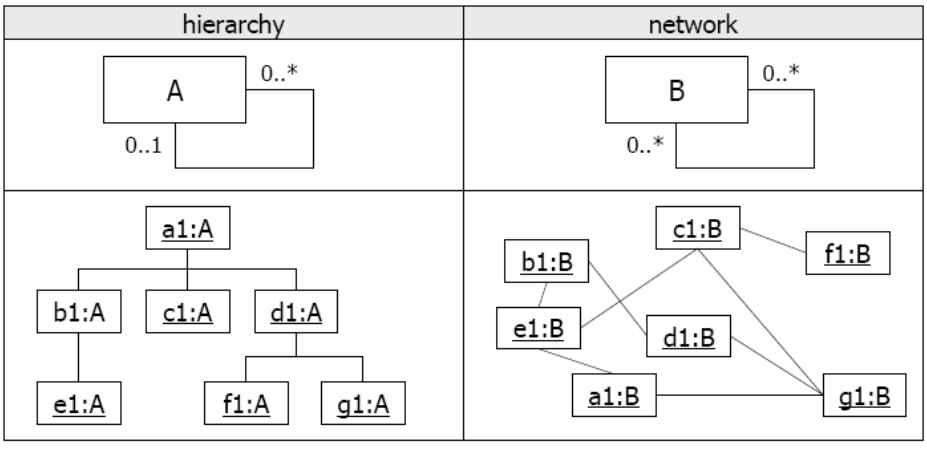
---

73) Co je to, když objekty jsou uspořádány do hierarchie nebo do sítě?

Průchodnost asociací:



Rozdíl mezi hierarchií a sítí:



---

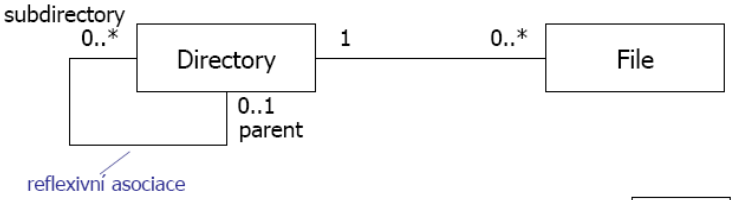
74) Co mají společného atributy a asociace?
lze je často zaměňovat
Atributy máme používat, když je na konci nedůležitá třída
Asociace pak tehdy, pokud je na konci důležitá třída, jejíž přítonost chceme zdůraznit.

---

75) Co je závislost?
Když se změna v dodavateli projeví také automaticky na klientovi, klient je závislý
Závislosti jsou znázorňovány jako tečkované šipky od klienta k dodavateli
Užití závislostí:
<<instantiate>> - klient je instancí dodavatele
<<use>> - klient používá dodavatele jako argument, návratovou hodnotu nebo jako prvek vlastní implementace
<<call>> - klientská operace volá dodavatelskou
<<parametr>> - dodavatel je argumentem
<<send>> - klient odesílá dodavatele
<<trace>> - klient je historickým vývojem dodavatele
<<substitute>> - klient může být za běhu nahrazen
<<refine>> - klient je další verzí dodavatele
<<derive>> - klient může být odvozen od dodavatele

---

76) Co je to reflexivní asociace?
Vazba sama na sebe

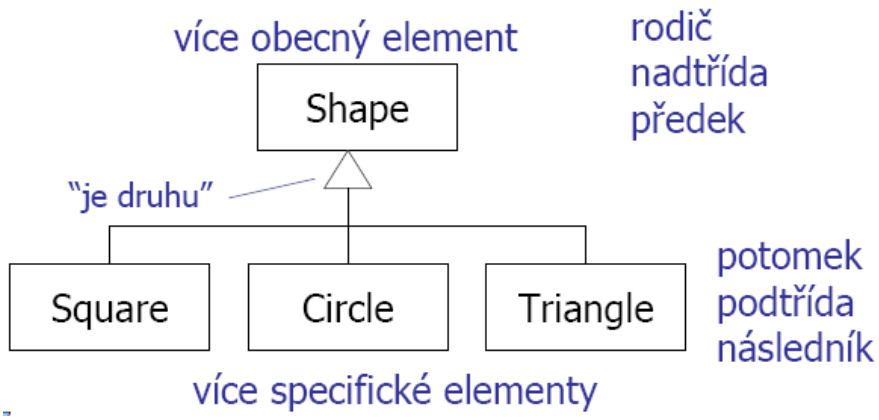


Toto je ukázka adresářové struktury. Která je hierarchická!!

10.Dědičnost a polymorfismus

---

77) Co je to zobecnění?
Realizace mezi obecnějším a přesněji specifikovaným. Konkrétnější předměty jsou důsledně konzistentní s obecnějšími přesměty. Zákon o nahraditelnosti říká, že obecnější předmět lze vždy nahradit konkrétnějším předmětem. Zobecnění se týká všech klasifikátorů a některých dalších modelovacích prvků. Hierarchie zobecnění lze tvořit postupným zobecňováním nebo postupným upřesňováním. Všechny předměty na stejné úrovni hierarchie by měly být na stejném stupni abstrakce.



---

78) K čemu dochází v relaci zobecnění mezi třídami?
K dědění tříd.

---

79) Co dědí potomek od svého předka?
Potomek dědí od svého předka: atributy, operace,relace a omezení

---

80) Co mohou potomci?

Potomci mohou přidávat novou charakteristiku, překrývat zděděné (abstraktní) operace. (Definuje jí se stejnou signaturou jako je operace předka)

---

81) Z jaké myšlenky vychází abstraktní operace?

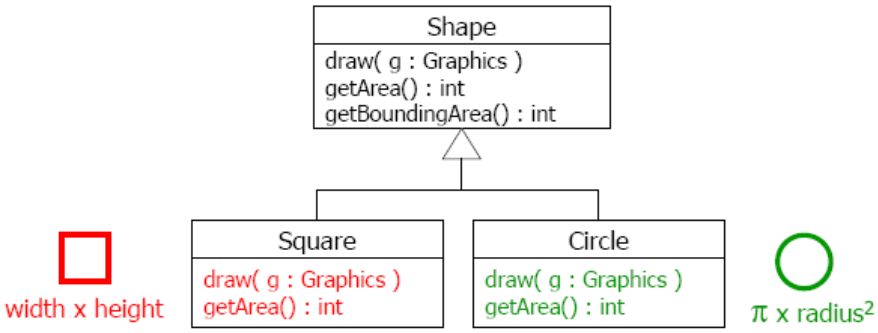
Abstraktní operace = nemá implementaci.

Slouží jako držitel prostoru a všechny podtřídy musí implementovat onu abstraktní funkci,ale tentokrát konkrétněji.

---

82) Co obsahuje abstraktní třída?

Alespoň jednu abstraktní operaci. Abstraktní třídy neumožňují tvorbu vlastních instancí.

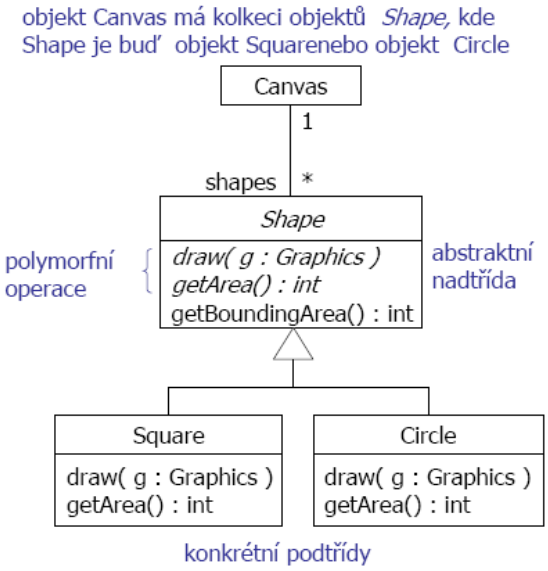


---

83) Co znamená polymorfismus?

Polymorfismus = mnohotvárnost. Abstraktní třídu nahradí jejími konkrétními potomky. Takové systémy jsou velmi flexibilní a rozšiřitelné. Polymorfní operace mají více implementací.

Pro rozšíření třídy jednoduše doplníme další potomky třídy.



11.Analytické balíčky

---

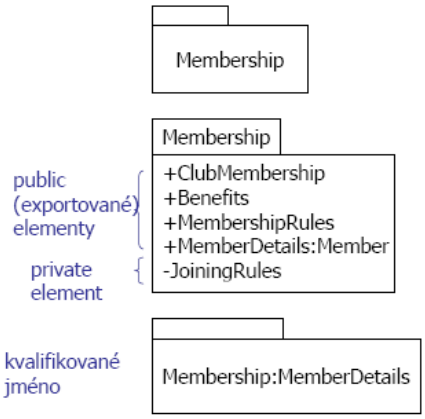
84) Jakým mechanismem je balíček?

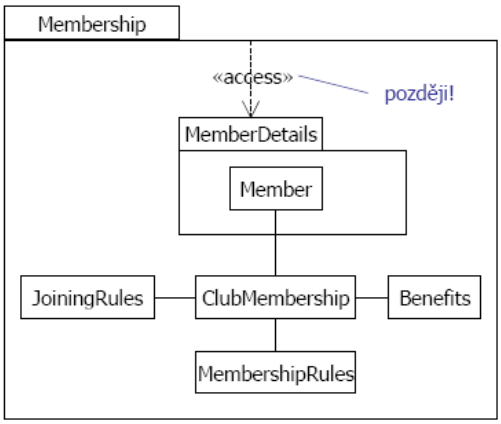
Kontejner a vlastník modelovaných prvků, má vlastní jmenný prostor a názvy objektů uvnitř něj musí být vždy jedinečné. Balíček je ve skutečnosti univerzálním mechanizmem uspořádání prvků a diagramů do skupin

- seskupují sémanticky příbuzné prvky
- vytvářejí sémantické hranice uvnitř modelu
- poskytují jednotky pro správu konfigurace
- poskytují jednotky pro souběžnou práci

-poskytují zapouzdřené jmenné prostory

Deklarace balíčků





Viditelnost balíčků je buďto:
(-) neviditelný, (+)public čili veřejný nebo (#) protected mode a to znamená, že je viditelný pouze děděným balíčkům.
Je doporučeno minimalizovat relace napříč balířky.

---

85) V kolika balíčcích může být jeden prvek?
Každý prvek je vlastněn jedním balíčkem. Balíčky vytvářejí hierarchie

---

86) Co mohou obsahovat analytické balíčky?
Případy užití
Analytické třídy
Realizace případů užití

---

87) K čemu slouží viditelnosti prvků v balíčku?
K řízení vzájemného provázání jednotlivých balíčků
- veřejná - prvky jsou dostupné dalším balíčkům
- soukromá - prvky jsou zcela skryty

Nepřechodnost balíčků: Když B dědí po A a C dědí po B, pak prvky balíčku A nevidí prvky balíčku C. Access je nepřechodný stereotyp a Import je přechodný stereotyp.

---

88) Znáte nějaké stereotypy u balíčků?
<<framework>> - balíček obsahující vzory
<<modelLibrary>> - balíček obsahující prvky určené pro užití jinými balíčky

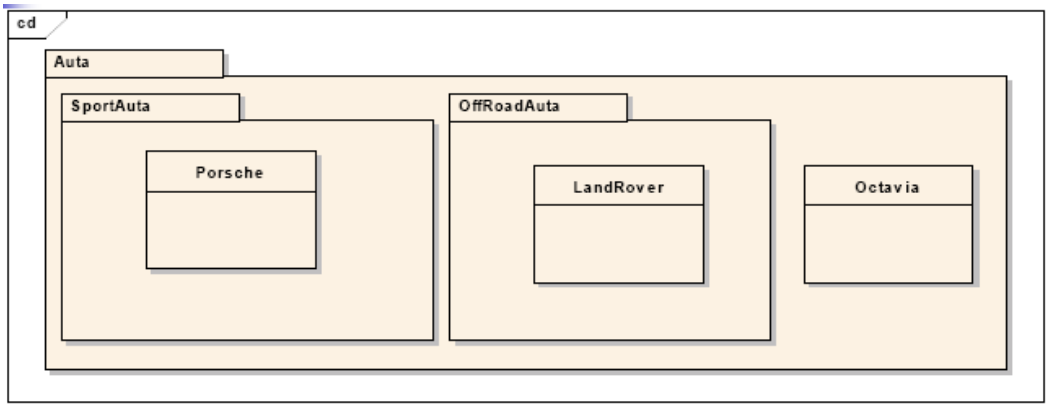
---

89) Co mají společného balíčky a jmenné prostory?
Balíček definuje zapouzdřený jmenný prostor. Při použití prvku z jiného balíčku musíme použít následující syntaxy:
Knihovna::Uživatelé::Knihovník
Přistupujeme přes „čtyřtečku“

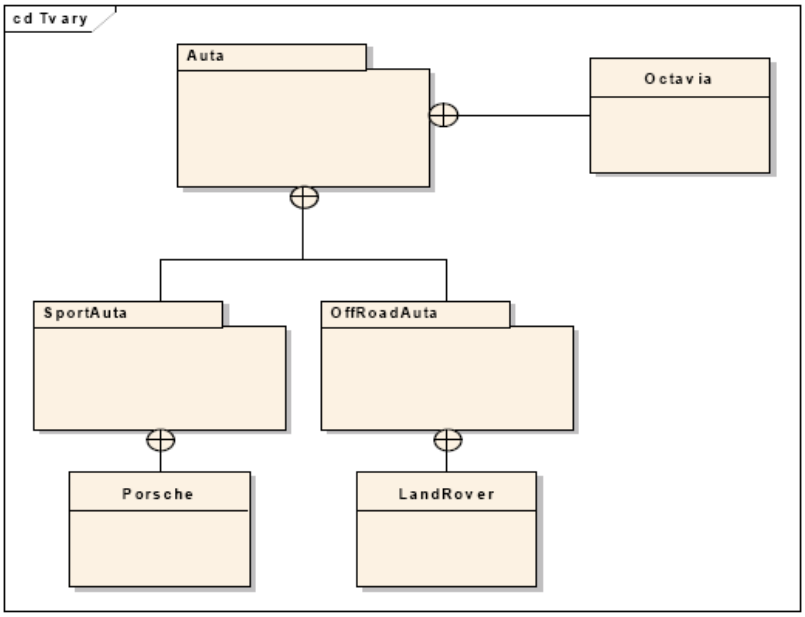
---

90) Lze balíčky vnořovat?
Ano
Vnitřní balíček vidí všechny veřejné členy vnějšího balíčku.
Vnější balíček nevidí žádné členy vnitřního balíčku, pokud na nich není explicitně závislý. Slouží to pro ukrytí implicitních vztahů
ve vnořených balíčcích.

Ukázka vnořených balíčků:



Nebo to lze zapsat takto: (symol kotvy = zdrojový objekt obsahuje cílový objekt)



---

91) Jaké relace můžou být mezi balíčky?
<<use>> - klientské prvky používají všechny veřejné složky dodavatele.
<<import>> - užívají všechny složky dodavatele a jmenný prostor je sloučen s prostorem klienta
<<access>> - všechny prvky dodavatele, ale jmenné prostory zůstávají odděleny
<<trace>> - klient je dalším vývojovým stupněm dodavatele. Obvykle se vztahuje na modely nikoliv na prvky
- historický vývoj jednoho balíčku
<<merge>> - veřejné prvky dodavatelského balíčku jsou sloučeny s klientským a využití: metamodelování. nesetkáme se s tím u OOP. (Rumbaugh)
<<model>> - úplný model UML

---

92) Lze zobecňovat balíčky?
Ano, podobá se to zobecňování tříd
Odvozené balíčky dědí prvky od svých předků a přidávají nové prvky a mohou také překrývat prvky předka.

---

93) Co je to architektonická analýza?
Dělí soudržné množiny analytických tříd na analytické balíčky a rozvrstvuje je podle sémantiky.
Pokouší se minimalizovat vzájemné vazby tím, že minimalizuje závislosti mezi balíčky a počet veřejných a chráněných prvků ve všech balíčcích.

---

94) Jak lze hledat analytické třídy?
Průzkum analytických tříd
- hledání sudržné skupiny úzce souvisejících tříd
- hledá se hierarchie dědičnosti
- 4 úrovně vazeb: dědičnost,kompozice,agregace,závislost
Průzkum případů užití
Maximalizace soudržnosti uvnitř balíčků a minimalizace závislostí mezi balíčky postupným upřesňováním modelu balíčku

- Přesouvání tříd mezi balíčky
- přidávání nových balíčků
- Odstraňování nepotřebných balíčků

---

95) V čem spočívá aktivita metodiky UP "Analýza případů užití"?
Spočívá v tvorbě realizací případů užití a je součástí dynamického pohledu na systém

12.Realizace případů užití

---

96) V čem spočívá realizace připadů užití?
Spolupráce instancí analytických tříd pro zajištění funkčních požadavků specifikovaných v případech užití

---

97) Kolik případl užití zachycuje jedna realizace?
Každá realizace případů užití zachycuje přesně jeden případ užití.

---

98) Z čeho se skládá realizace připadů užití?
Realizace se skládá z:
- diagram analytických tříd, které "vypravují" příběh o případech užití
- diagram iterakcí které znázorňují spolupráci skupin objektů pro dosažení chování případů užití
- speciální požadavky
- upřesňování případů užití, původní záměr si často vyžádá změnu případů užití.

---

99) Co by měly poskytovat diagramy analytických tříd?
Diagram analytických tříd "vypravují" příběh o případech užití

---

100) Co jsou to interakce?
Interakce jsou jednotkami chování kontextového klasifikátoru. Mohou používat libovolné funkce kontextového klasifikátoru. V realizaci případů užití je kontextový klasifikátor případem užití

---

101) Může realizace případl užití změnit případy užití?
Upřesňování případů užití, původní záměr si často vyžádá změnu případů užití.

---

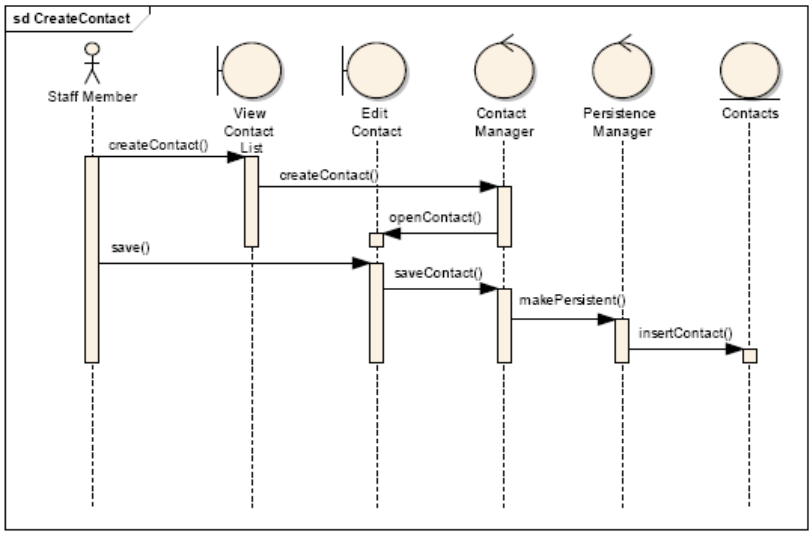
102) Co zastupuje "čára života"?
Účastníka iterakce, tedy způsob, jímž se instance podílí na interakci. Všechny čáry života mohou mít název, typ a selektor.

---

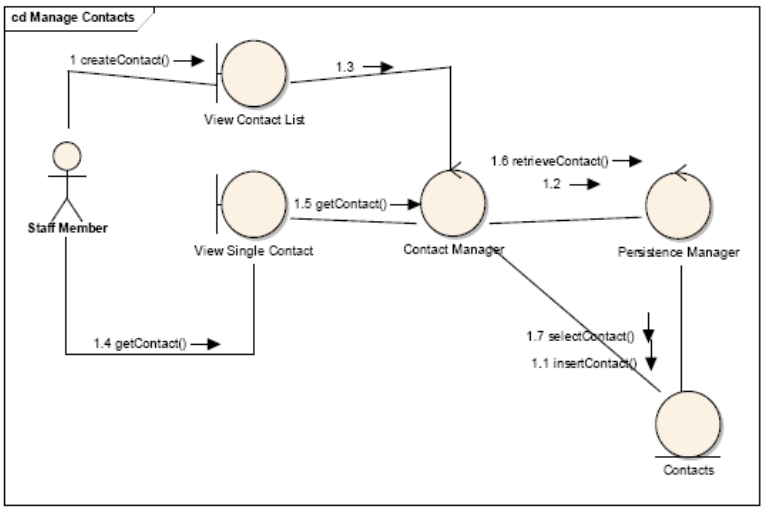
103) Co představuje zpráva mezi čárami života?
104) Jaké chování mají synchronní, asynchronní a návratové zprávy?
Zastupuje určitý druh komunikace mezi dvěma čárami života.
- synchronní zpráva - zavřený hrot šipky
- asynchronní zpráva - otevřený hrot šipky
- návrat zprávy - otevřený hort šipky, čerchovaná čára
- vytvářecí zpráva - otevřený hrot, plná čára, stereotyp <<create>>
- mazací zpráva - otevřený hrot šipky, plná, <<destroy>>
- zpráva o nalezení - otevřený hrot, linka začíná vyplNěnou kružnicí
- zpráva o ztrátě - otevřený hrot, linka končí vyplněnou kružnicí

---

105) Jaké jsou diagramy interakce?
a) sekvenční diagramy - časová posloupnost odeslaných zpráv



b) komunikační diagramy - strukturální vztahy mezi objekty



c) zjednodušený diagram interakcí - zdůrazňuje vztahy mezi interakcemi

d) diagram časování - časov aspekty iterakcí

---

106) Co znázorňují sekvenční diagramy?

Čas běžící odshoru dolů, čáry života jsou vedeny zleva doprava, čáry jsou prezentovány jako čerchované svislé ohony, jež naznačujídélky života objektu, třídy a jejich zapojení do posílání zpráv.

---

107) Co jsou fragmenty a operátory v sekvenčních diagramech?

Sekvenční diagram lze rozdělit na kombinovaný fragment. Každý fragment má po jednom operátoru a obsahuje kontrolní podmínky průchodů. Operátor určuje chování operandu

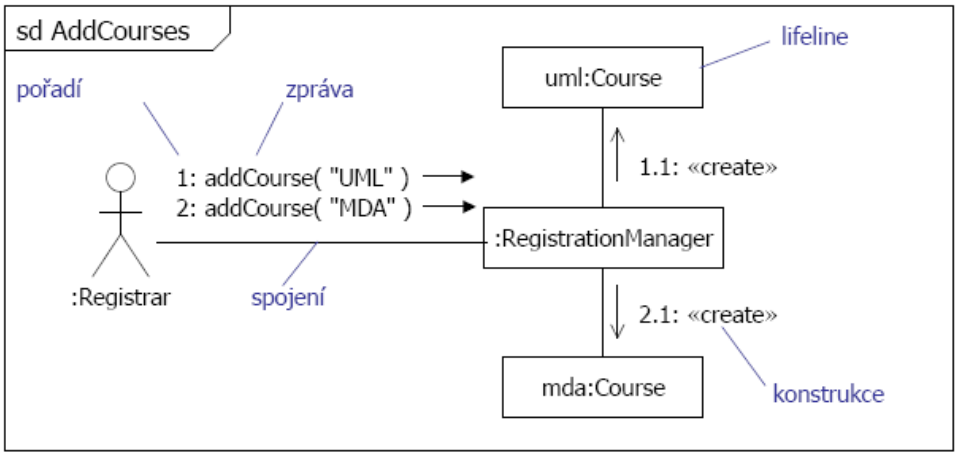
Příklady operátorů:

operator	long name	sémantika
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

---

108) Co znázorňují komunikační diagramy?

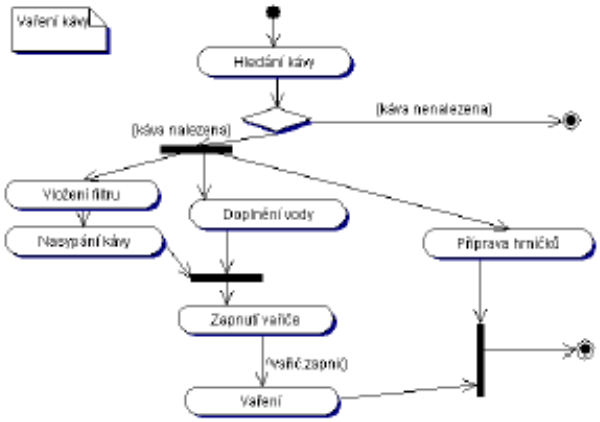
Zdůrazňují strukturální aspekty interakce, zprávy mají pořadová čísla a jsou číslovány hierarchicky podle vnoření aktivace. Ukázka komunikačního diagramu:



14.Diagramy aktivit

---

109) Co jsou diagramy aktivit?
OO vývojové diagramy. Díky nim lze proces modelovat jako aktivitu, která se skládá z kolekce uzlů spojených hranami.
V UML 1 byly pouze zvláštním případem stavových automatů
Objektově orientované diagramy aktivit používané při modelování všech typů procesů
Dobrý diagram aktivit popisuje jeden konkrétní aspekt chování systému
v UML2 používají diagramy aktivit sémantiku petriho sítí



---

110) K jakému prvku lze připojit diagram aktivit?

---

111) Jakými způsoby se použijí diagramy aktivit v analýze?

---

112) Z čeho se skládají aktivity?
Kategorie uzlů
Kategorie hran
Tokeny

---

113) Co jsou uzly, hrany a tokeny?
Oddíly aktivit

---

114) Vyjmenujte akční uzly?
Zastupují samostatné jednotky, jež jsou v rámci aktivity nedělitelné. Na vstupních hranách mají tokeny
Po dokončení nabíuejí simultálně na svých výstupech své tokeny
- akční uzly volání - volání aktivity,chování, operace
- akční uzel Odeslat signál
- akční uzel Přijmout událost

- akční uzel Přijmout časovou událost, je spuštět po splnění časové podmíny. Události v čase,v okamžiku,trvání

---

115) Vyjmenujte řídící uzly?

typy

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

---

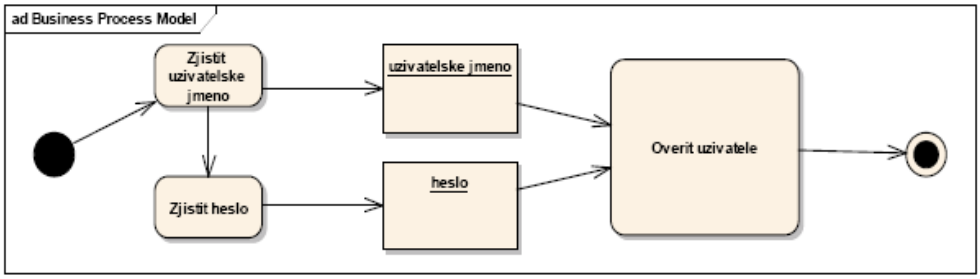
116) Co zastupují objektové uzly?

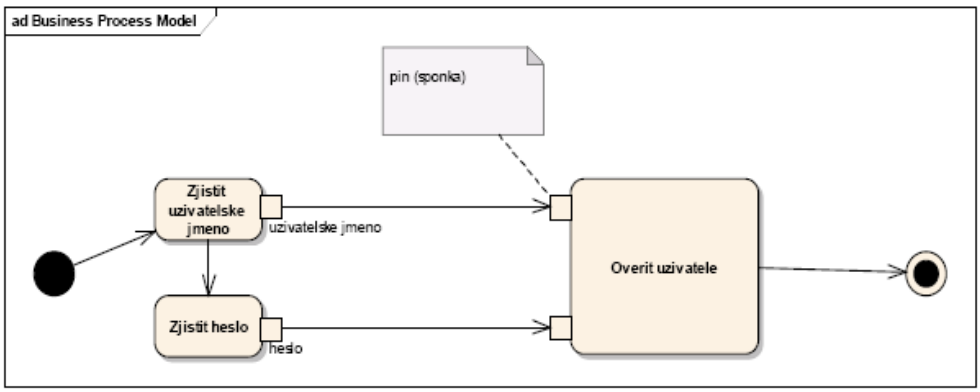
Zastupuji objekty použité v rámci dotyčné aktivity

---

117) Co jsou sponky?

Objektové uzly zastupující jeden vstup nebo jeden výstup akce nebo aktivity.





---

118) Co jsou spojky?

Používají se k přerušení dlouhých a složitých hran. Ke zjednodušení modelovaného schématu. Ale může být složité je sledovat, kde končí a kde začínají. Vtupní a výstupní spojka.

---

119) Jak se ošetřují výjimky?

Exception handling.

Vytvořen objekt vyjímky a zachytávání vyjímek. V diagramech aktivit můžeme chyby ošetřovat pomocí:

a)sponek s vyjímkami

b)chráněnými uzly

c)chybovými rutinami

---

120) Co zastupují přídavné uzly?
Oblasti, které jsou iterativně vykonány pro každý objekt v kolekci.
---
121) Co jsou signály a jak se znázorňují?
Signály zastupují informace
---
122) Co je to prodění a jak se znázorňuje?
Akce, která probíhá neustále dokud se přijímají a nabízejí tokeny.
