UML, alea iacta est!

Miroslav BENEŠOVSKÝ[1], Karel RICHTA[2]

[1]*BenSoft s.r.o*
Brusy 285,66484 Zastávka u Brna
mben@ics.muni.cz
[2]*Katedra softwarového inženýrství, MFF UK Praha*
Malostranské nám. 25, 118 00 Praha
richta@ksi.ms.mff.cuni.cz

Abstrakt. Z hlediska současného stavu v oblasti vývoje SW se zdá, že kostky UML byly již vrženy. Tutoriál se zabývá rozborem, kam dopadly a kolik je výsledný součet. Obsahuje dvě části - teoretický rozbor aktuálního stavu UML a rozbor praktického využití UML. Problematika je ilustrovaná zejména na příkladech.

Klíčová slova: UML, UP, RUP, OCL

1 Úvod

Unifikovaný modelovací jazyk UML se prosadil jako standard - verze 1.4.2 byla přijata jako standard ISO/IEC 19501:2005(E) [4]. Zdá se tedy, že kostky UML byly již vrženy. Cílem tohoto příspěvku je shrnout, kam všude dopadly a jak se v praxi UML využívá. UML se ovšem stále vyvíjí, v současnosti je aktuální verze 2.1.2 [7]. Naším cílem je proto rovněž vysvětlit, jaké novinky přicházejí a k čemu jsou dobré.

V prvé části tutoriálu uvedeme přehled UML – přehled jednotlivých typů diagramů, bez nároku na úplnost, ale spíše ilustrativní příklady použití. Ve druhé části se pak zaměříme na praktické aspekty používání UML, zejména těch nejdůležitějších artefaktů.

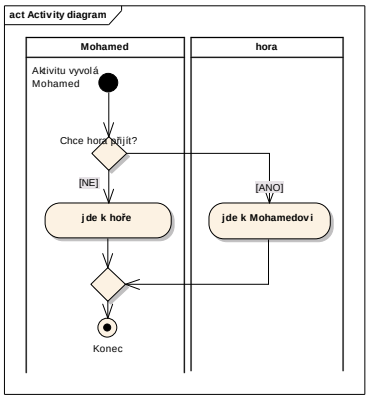
UML je notace, nezbytným předpokladem praktického použití je metodika, která vytváření artefaktů v UML vyžaduje a vede vývojový tým k používání tohoto prostředku a porozumění jeho sémantiky. Proto se v praktické části budeme mimo jiné zabývat vztahem metodiky a UML.

1.1 Proč opět UML?

Na konferenci DATAKON se již o unifikovaném modelovacím jazyku UML mluvilo několikrát. V roce 1999 se konala první zvaná přednáška „Modelovací jazyk UML“ [6], v ročníku 2002 byl uveden tutoriál o UML s názvem „UML – teorie a praxe“ [4]. Proč je tedy znovu UML na programu konference DATAKON?

Zdá se, že nastal čas, kdy lidé okolo tvorby programů postupně akceptovali existenci tohoto prostředku a začínají ho skutečně používat, nebo jsou k jeho používání nuceni. Pomocí UML komunikujeme s širokou škálou lidí, počínaje zákazníky, ale zejména se dohadujeme mezi sebou - IT komunita se musí umět domluvit. UML může posloužit jako komunikační prostředek. Např. chceme vysvětlit kolegovi přísloví „*Pokud nechce hora k Mohamedovi, musí Mohamed k hoře.*“. K popisu této aktivity můžeme použít diagram aktivity UML (viz Obr. 1), protože se na to hodí. To je jeden z cílů tohoto tutoriálu, probrat

znovu základní rysy UML, zdůraznit, v jaké situaci se ten který artefakt hodí. Současně upozornit na nové rysy a jejich zamýšlený účel.



Obr. 1: Příklad diagramu activity

Tento text slouží jako pomůcka pro posluchače tutoriálu. Obsahuje popis notace UML ilustrovaný na příkladech. Teoretické a praktické problémy používání UML budou obsahem přednášky, zde jsou shrnuta probíraná témata.

1.2 Modelový příklad - Hotelový rezervační systém

Uvažme jako ilustrační příklad projekt hotelového rezervačního systému (HRS), na kterém budeme ukazovat možnosti jednotlivých diagramů. Vize projektu říká, že zákazníkem je společnost provozující řetězec ubytovacích zařízení, která vznikla původně jako rodinná firma. Vzhledem k tomu, že společnost úspěšně expanduje a rozšiřuje svojí působnost, chtěla by nahradit stávající evidenci rezervací, která není vyhovující a způsobuje velké finanční náklady s tím spojené. V současné době se evidují rezervace pro hotelové pokoje lokálně v aplikaci MS Excel. Tyto soubory jsou měsíčně zasílány na vedení firmy, kde jsou z těchto dat vytvářeny reporty pro obchodní potřeby.

Nový systém je určen pro řízení rezervací ubytování, webový přístup pro prohlížení ubytovacích zařízení a pokojů, sledování aktuálního stavu rezervací a vytváření nových rezervací a koordinaci pokojů. Zákazník předal jako podklady pro zpracování úvodní studie projektu dokumenty, které obsahují základní informace o zákazníkovi a o současném stavu, požadavky na funkčnost systému, které jsou roztříděny do tří kategorií – nezbytné, významné a návazné a požadavky na dostupnosti a výkonnost.

Dle zadání by předmětem zakázky měla být dodávka tří komponent: aplikace pro recepci v hotelech (HotelApp), webové aplikace pro přístup zákazníků ze sítě (WebApp) a kiosk, sloužící pro zveřejňování lokálních informací – restaurace, kulturní akce apod. (KioskApp). Dle popisu v dokumentaci není komponenta KioskApp závislá na ostatních aplikacích a disponuje vlastními procesy, které mohou být dosti rozsáhlé. Detailní specifikace může znamenat prodloužení nasazení očekávané aplikace, což pro zákazníka přináší další finanční ztráty. Rezervace hotelových pokojů a práce s ubytovacími objekty je pro daný typ zákazníka klíčová a implementací omezené části požadovaného systému se můžeme vyhnout případným ztrátám, které můžou způsobit rizika uvedená v dokumentaci.

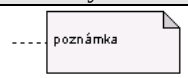
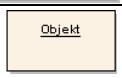
V důsledku výše uvedeného navrhneme zákazníkovi rozdělit implementaci do dvou fází – nejprve řešení aplikací HotelApp a WebApp, poté aplikace KioskApp. V tomto textu budou probírány pouze prvé dvě aplikace.

2 UML – teorie

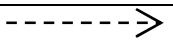
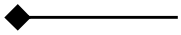
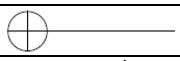
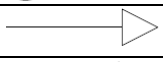
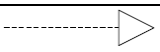
V této kapitole stručně shrneme základní rysy UML. Specifikace UML obsahuje 4 základní části, které popisují *infrastrukturu* (zhruba: základní elementy a vztahy), *superstrukturu* (zhruba: diagramy a jejich význam), *výměnný formát* (jak zápis v UML exportovat a importovat) a definici *jazyka OCL* (Object Constraint Language), ve kterém lze formálně přesně zapisovat integritní omezení modelů. Jazyk UML je definován pomocí modelu v UML – metamodel UML je zapsán pomocí diagramů tříd UML opatřených popisem sémantiky a doplněn formálním vyjádření sémantiky v OCL.

2.1 Elementy a vztahy

Každý model je dokumentován sadou pohledů. Model v UML je dokumentován sadou diagramů, které dokumentují určité rysu modelu. Každý diagram je sestaven z jistých elementů a vztahů mezi nimi. Obecně se v UML připouští, aby v každém modelu byl použít libovolný element. Ne všechny kombinace jsou ale smysluplné, vždy určitá kombinace elementů a vztahů představuje superstrukturu diagramu jistého typu. Některé elementy a vztahy jsou použitelné obecně – viz Tab. 1 a Tab. 2.

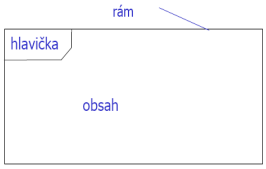
Element (prvek)	UML syntaxe	UML sémantika
poznámka		doplňující podrobnější popis elementu nebo diagramu
Balík (package)		logická skupina elementů
objekt (instance)		konkrétní objekt
Třída		třída objektů, které mají něco společného

Tab. 1: Obecné elementy UML

Vztah (relationship)	UML syntaxe	UML sémantika
vztah (asociace)		vztah mezi elementy (symetrický)
souvislost		související elementy (připojení poznámky)
závislost		jeden element závisí na jiném elementu
agregace		Element obsahuje jiný element (celek-část)
kompozice		silnější agregace (stejný životní cyklus)
obsahuje		zdrojový element obsahuje cílový element
generalizace		zdrojový element je specializací cílového
realizace		zdrojový element je realizací cílového

Tab. 2: Obecné vztahy v UML

Diagram v UML má syntaxi dle obrázku Obr. 2, kde hlavička popisuje druh diagramu, jeho jméno a případné parametry. Diagram obsahuje elementy a vztahy mezi nimi, typem diagramu je určena sada elementů a vztahů, které mají v daném pohledu definován význam.

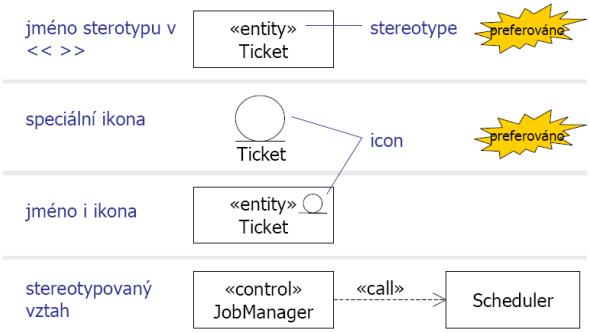


Obr. 2: Syntaxe diagramu UML

2.2 Ornamenty elementů

Pro každý typ elementu existuje textová specifikace syntaxe a sémantiky. Každý element má základní podobu, ke které je možno doplnit další ornamenty (adornments). Ornamenty jsou zobrazovány jen pro upřesnění vlastností elementu. Základní sadu elementů lze rozšiřovat pomocí stereotypů, omezení (constraints) a příznaků (tags).

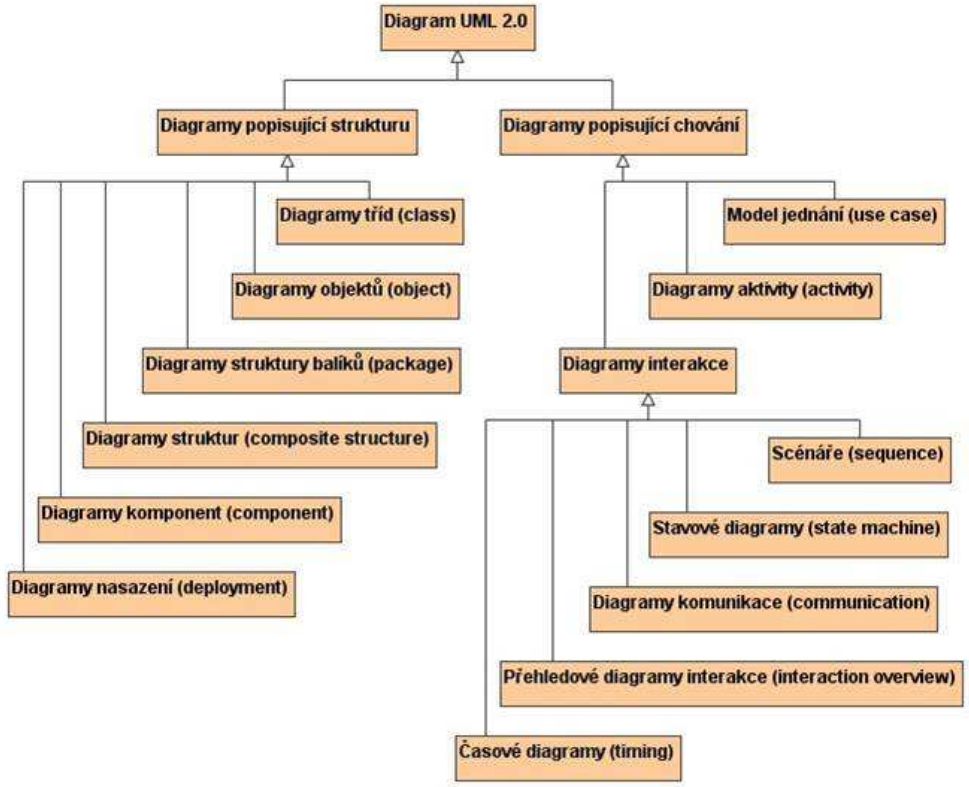
Stereotyp umožňuje definovat nový element UML na základě existujícího, sémantiku si definujeme sami. Zapisuje se «stereotypeName». Omezení rozšiřují sémantiku elementu o nová pravidla, zapisují se do složených závorek: { pravidlo }. Příznaky (Tagged values) umožňují přidat novou ad-hoc informaci (atribut) k elementu, zapisují se: { tag1 = value1, tag2 = value2 … }.



Obr. 3: Možné způsoby zápisu stereotypů

2.3 Přehled diagramů

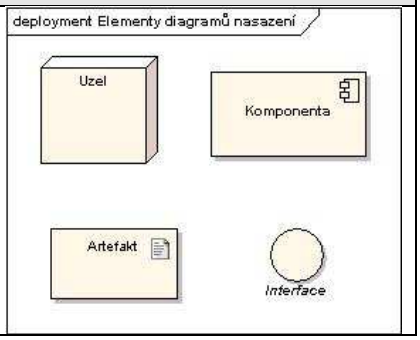
Základní vodítko pro taxonomii diagramů představuje hledisko, zda popisují strukturu nebo chování systému – zda slouží pro dokumentaci statického, nebo dynamického pohledu na dokumentovaný systém. Struktura je na nejvyšší úrovni dokumentována diagramy nasazení – ty popisují nasazení komponent na výpočetní prostředky. Další stupeň představují diagramy komponent – popisují, jak se nasazované komponenty sestaví. Komponenty se pak skládají z balíků tříd, které je implementují. Chování se na nejvyšší úrovni popisuje diagramu případů užití, poté se dokumentují aktivity a scénáře jednotlivých případů užití. Popis dynamiky doplňují další dynamické popisy – stavové modely, časové modely, příp. přehledy interakce. Celkový přehled (super)struktury diagramů dokumentuje (pomocí diagramu tříd) obrázek Obr. 4.



Obr. 4: Přehled diagramů UML

2.4 Diagramy nasazení (Deployment Diagrams)

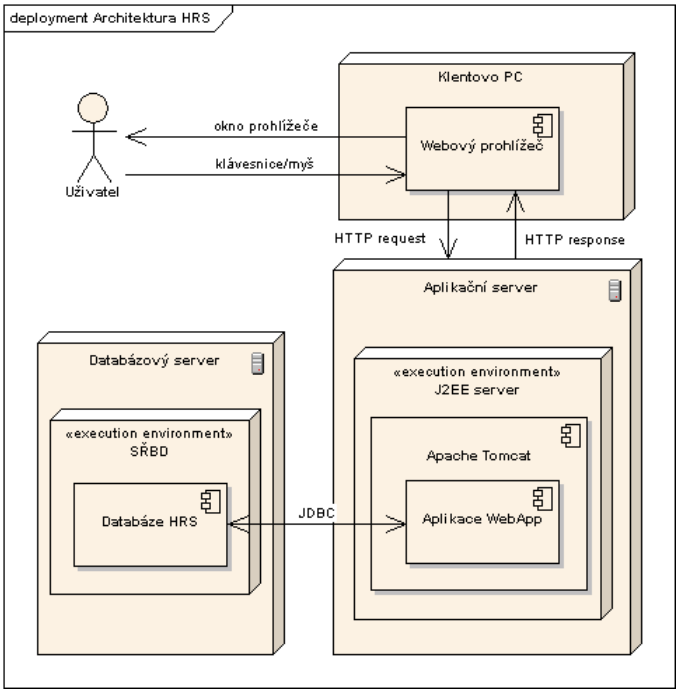
Pro návrh a dokumentaci architektury systémů používáme diagramy nasazení. Jejich základní elementy shrnuje tabulka Tab. 3. V tomto pohledu se uvádějí komponenty, které jsou nutné pro běh systému při jeho nasazení.

Element (prvek)	UML sémantika	UML syntaxe
uzel	znázorňuje výpočetní prostředek	
komponenta	logická část celku, zde představuje část potřebnou pro běh systému	
artefakt	určitý objekt (soubor, dokument, …)	
rozhranní (interface)	rozhraní poskytované třídou či komponentou	

Tab. 3: Základní elementy diagramů nasazení

Architektura HRS

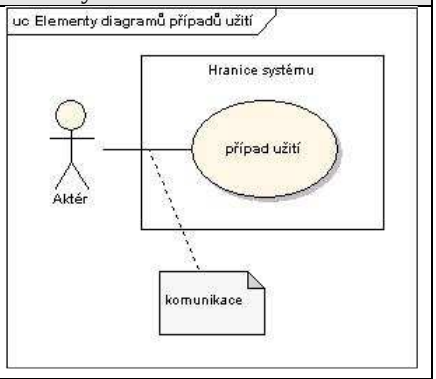
Jako příklad diagramu nasazení lze použít hrubý návrh architektury systému HRS, který je na obrázku Obr. 5. Poslouží nám pro základní orientaci v navrhované struktuře systému a jeho částech.



Obr. 5: Architektura HRS

2.5 Model jednání, diagramy případů užití (Use Case Model)

Základním pohledem dokumentujícím funkčnost systému je model jednání. Model jednání zahrnuje diagramy případů užití a jejich popis. Přehled základních elementů diagramů případů užití obsahuje tabulka Tab. 4.

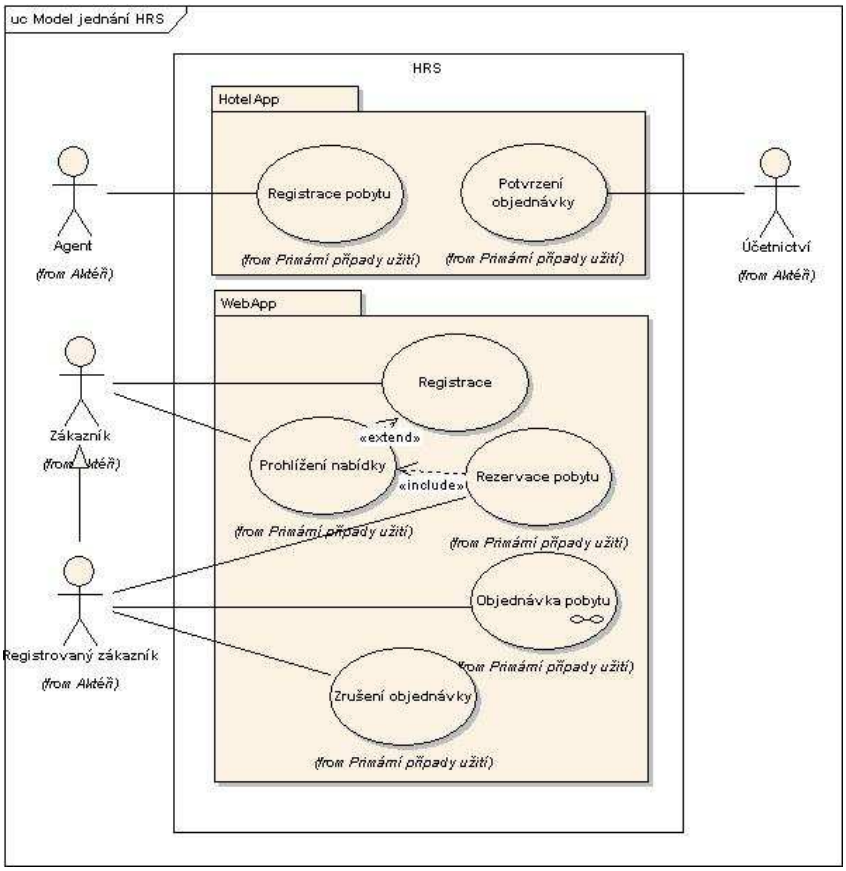
Element (prvek)	UML sémantika	UML syntaxe
aktér (actor)	znázorňuje uživatelskou roli, či spolupracující výpočetní systém	
hranice systému (systém boundary)	vyznačuje oblast zodpovědnosti popisovaného systému	
případ užití (use case)	dokumentuje možnost využít systém k realizaci služby	
komunikace	vyznačuje, který aktér může danou událost vyvolat – představuje komunikační kanál mezi aktérem a systémem	

Tab. 4: Elementy diagramů případů užití

Případy užití dokumentují požadované služby systému, jejich podrobný popis pak představu o požadovaných pracovních postupech. Tento popis se zpravidla nejprve vytváří v přirozeném jazyce (příp. strukturovaně), později se může doplnit rozmanitými diagramy.

Model jednání HRS

Zabývejme se požadavky na systém HRS. Budeme se snažit odhalit aktéry budoucí aplikace a zachytit pomocí diagramu případů užití. Základní role jsou „agent“ a „zákazník“. Agent je zaměstnanec společnosti, který eviduje nabídky pobytů, vystavuje je zákazníkům a realizuje akce „příjezd“ a „odjezd“. Tyto služby mu zajistí komponenta HotelApp. Zákazníci využívají komponentu WebApp, kde si mohou prohlížet nabídku pobytů. Pokud si ovšem chtějí pobyty objednat, musí se registrovat. Tato podmínka je nutná pro získání a uchování kontaktu na zákazníka.



Obr. 6: Model jednání HRS

Skutečný model jednání systému HRS musí být doplněn o popisy jednotlivých případů užití. Např. případ užití „Registrace“ může být popsán následovně (struktura popisu není v UML předepsána):

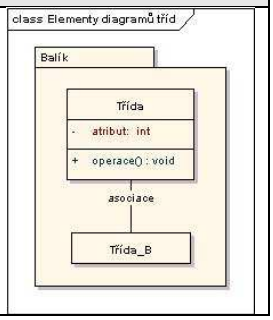
UCx: Registrace zákazníka

Předpoklad: Zákazník není registrován
Scénář: Zadání potřebných údajů, ověření zadaných údajů, záznam údajů do seznamu registrovaných zákazníků

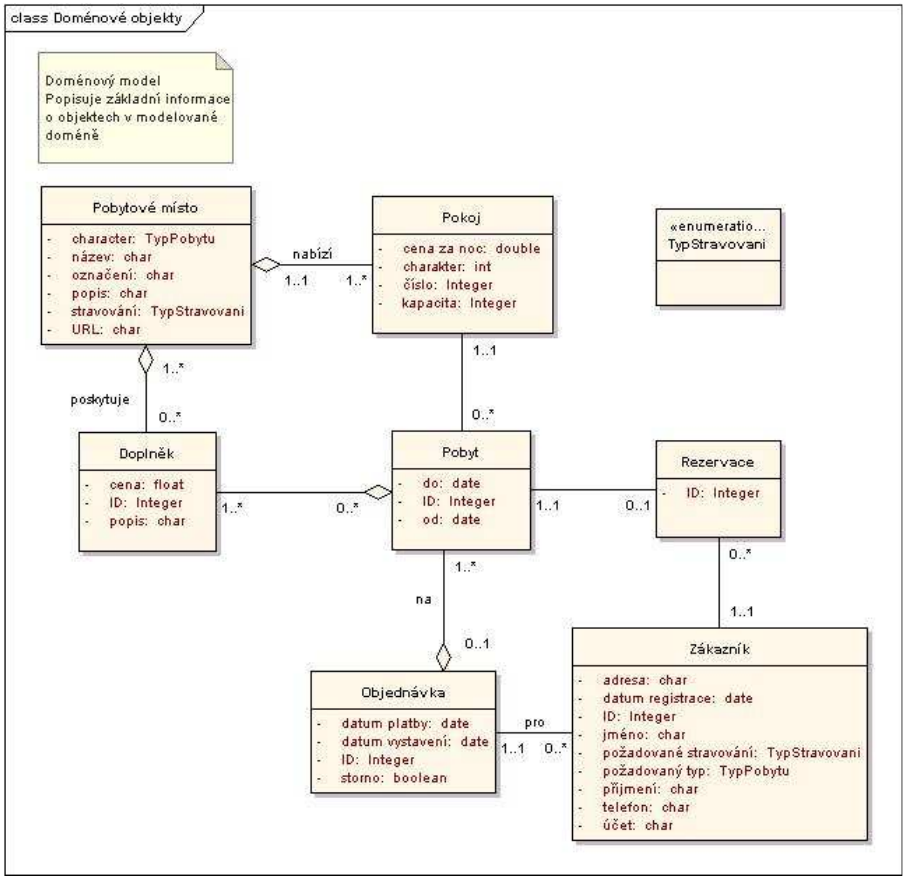
Následná podmínka: Zákazník je registrován.

2.6 Model tříd

Statický pohled na popisovaný systém vyjadřujeme v UML pomocí diagramů tříd, diagramů kompositní struktury, příp. diagramů komponent, či nasazení.

Element (prvek)	UML sémantika	UML syntaxe
třída (class)	dokumentuje entitu – datový typ, příp. jeho atributy (vlastnosti) a operace (chování)	
Balík (package)	umožňuje vyznačit logickou skupinu tříd	
asociace (vztah)	dokumentuje možnost existence vztahu mezi třídami	

Tab. 5: Elementy diagramů tříd



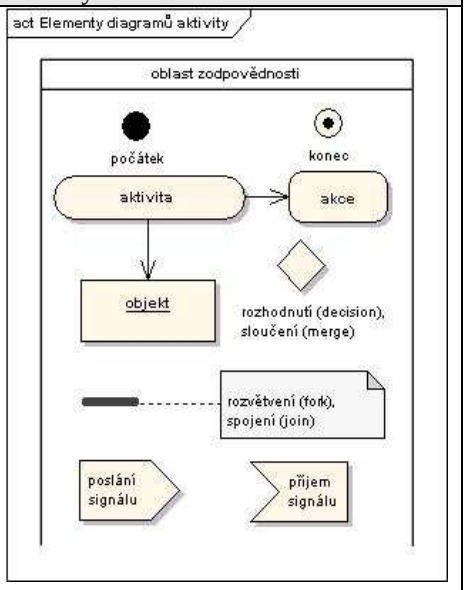
Obr. 7: Konceptuální model dat HRS

Konceptuální model dat pro HRS

Pro zajištění výše uvedených služeb HRS je zapotřebí evidovat data o pobytech a zákaznících. Tato data jsou vytvářena a zpracována pomocí služeb systému. Jejich strukturu můžeme zachytit diagramem tříd. Zatím pomocí tohoto diagramu zachytíme doménový model – modelujeme data dané domény (viz Obr. 7).

2.7 Diagramy aktivity

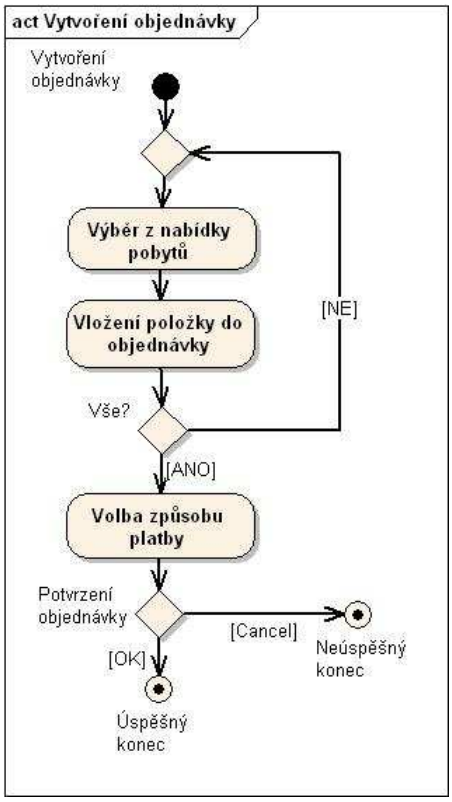
Chování se v UML popisuje různými typy diagramů, které mohou zachytit podstatné rysy dané činnosti. Pokud potřebujeme dokumentovat činnost či proces, který se skládá z posloupnosti navazujících kroků (jednodušších činností), je vhodné použít diagramy aktivity. Elementy diagramů aktivity popisuje tabulka Tab. 6.

Element (prvek)	UML sémantika	UML syntaxe
aktivita, proces (activity)	dokumentuje činnost nebo proces, který je zapotřebí vykonat	
akce (action)	dále nedělitelná činnost	
přechod (control flow)	umožňuje vyznačit návaznost aktivit, předání řízení po skončení aktivity	
datový tok (data flow)	dokumentuje produkované nebo konzumované objekty	
objekt	dokumentuje produkovaný nebo konzumovaný objekt (instanci)	
počátek	dokumentuje počátek aktivity	
konec	dokumentuje konec aktivit	
rozhodnutí (decision), sloučení (merge)	dokumentuje možnost rozvětvení dle zadané podmínky – větve jsou disjunktní	
rozvětvení (fork), spojení (join)	dokumentuje možnost paralelních činností, které se synchronizují při spojení	
poslání signálu (send), příjem signálu (receive)	dokumentuje možnost poslání signálu vně, či příjem signálu zvenku	
organizační jednotka (swimlane)	vymezuje oblast zodpovědnosti	

Tab. 6: Elementy diagramů aktivity

Aktivita při vytváření objednávky

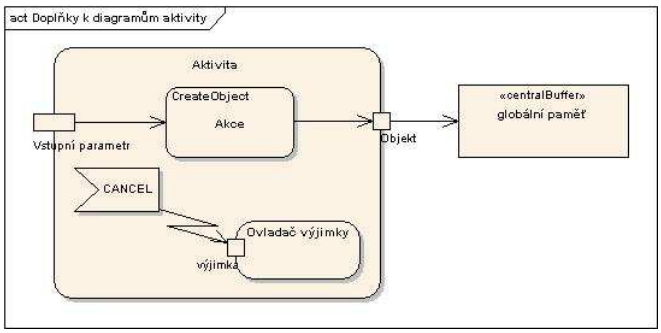
Pokusme se použití diagramů aktivity ilustrovat popisem jednoho případu užití systému HRS – vytvoření objednávky zákazníkem. Zamýšlíme se nad aktivitou, kterou musí registrovaný zákazník při vytváření objednávky absolvovat – viz obrázek Obr. 8.



Obr. 8: Diagram aktivity při vytváření objednávky

2.8 Doplňky k diagramům aktivity

Ve verzi UML 2 byly diagramy aktivity doplněny o koncept akcí – akce je primitivní, dále nedělitelná aktivita, ze kterých se skládají složitější aktivity.



Obr. 9: Doplňky k diagramům aktivity

Existuje předdefinovaná sada akcí – jakýsi virtuální UML stroj. Pokud všechny diagramy aktivity vybudujeme nad touto sadou, měly by být mechanicky interpretovatelné. Další novinkou jsou parametry aktivit a akcí. Rovněž lze vyjádřit přerušení aktivity a její zpracování ovladačem výjimky. Rozmanité problémy diagramů aktivity budu probírány na přednášce.

2.9 Stavové diagramy

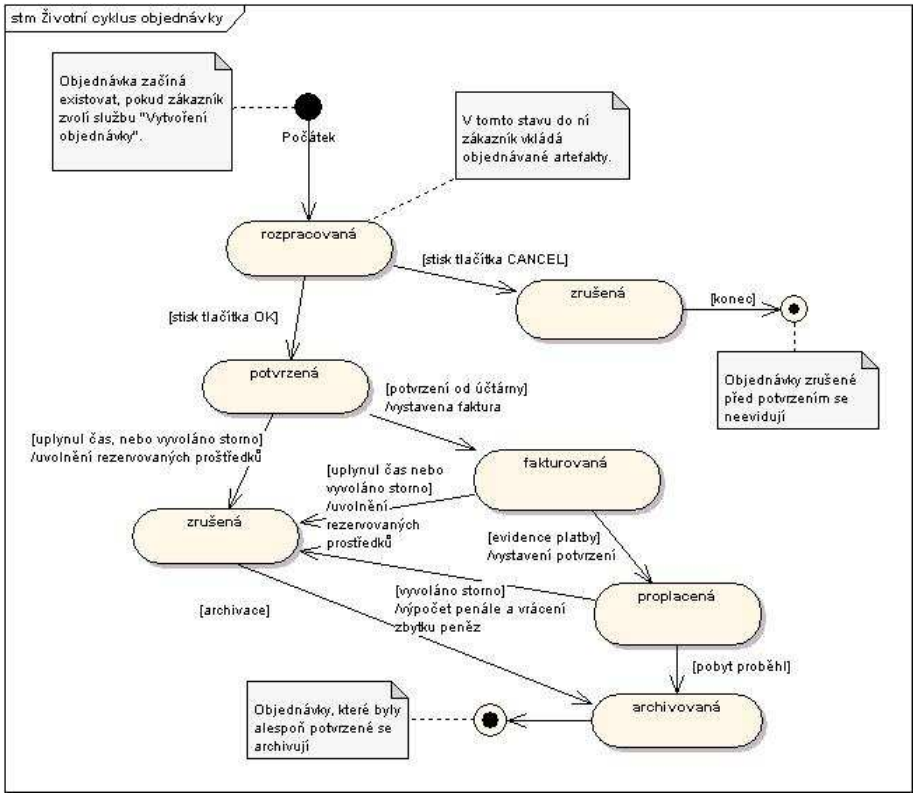
Diagramy aktivity jsou synchronní – následující činnost navazuje dokončením činnosti předchozí. Pro popis asynchronního chování řízeného událostmi zvenku se v UML používají stavové diagramy (nazývané v nové verzi UML stavové automaty – state machines). Často tak dokumentujeme životní cyklus nějakého objektu či komponenty, který závisí na sledu vnějších událostí. Elementy stavových diagramů popisuje tabulka Tab. 7: Elementy stavových diagramů.

Element (prvek)	UML sémantika	UML syntaxe
stav (state)	dokumentuje stav objektu nebo procesu, kdy se čeká na nějakou událost	
přechod (transition)	umožňuje vyznačit návaznost stavů po příchodu události	
počátek	dokumentuje počátek aktivity	
konec	dokumentuje konec aktivit	
vstupní bod (entry point), výstupní bod (exit point)	dokumentuje vstupní a výstupní bodu stavů	
historie	dokumentuje místa, kam se lze v historii vracet	
synchronizace	dokumentuje místa pro synchronizaci	

Tab. 7: Elementy stavových diagramů

Diagram životního cyklu objednávky

Z diagramu aktivity při vytváření objednávky je zřejmé, že objednávka nabývá v různých okamžicích různých stavů. Podle stavu objednávky ji lze např. potvrdit, zaplatit, zrušit. To vyjádříme stavovým diagramem na obrázku Obr. 10.



Obr. 10: Diagram stavů objednávky v HRS

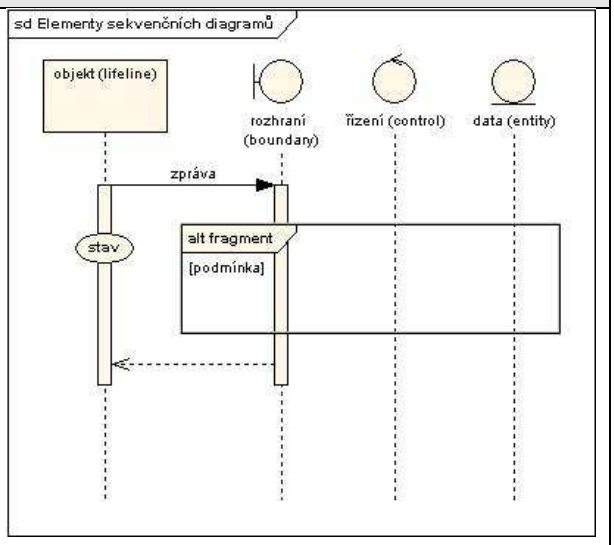
Přechody ve stavových diagramech by vždy měla způsobit nějaká vnější událost. Pokud tomu tak není, nejedná se o stav, ale aktivitu. Ta po dokončení přenáší řízení na aktivitu jinou. Přechod je obecně definován vstupní událostí – stimulem, který přechod způsobil. Akceptování stimulu lze vázat na platnost podmínky. Součástí vstupní události mohou být vstupní data. Reakcí na aktivaci přechodu jsou výstupní signály – aktivity, které jsou přechodem vyvolány. Obecně je přechod ve stavovém modelu ohodnocen následující specifikací:

[podmínka]událost(parametry události)/aktivity vyvolané přechodem

Zjednodušení zápisu umožňují tzv. vstupní, interní a výstupní aktivity, které jsou zahrnuty do popisu stavu a označeny návěštím enter, exit a do jako (pseudo)událostí. Stavové diagramy umožňují rovněž vyznačit historii – místa, do kterých se lze později vrátit. Stavové diagramy mohou samozřejmě být hierarchické – stav na vyšší úrovni zakrývá podrobnosti detailního chování.

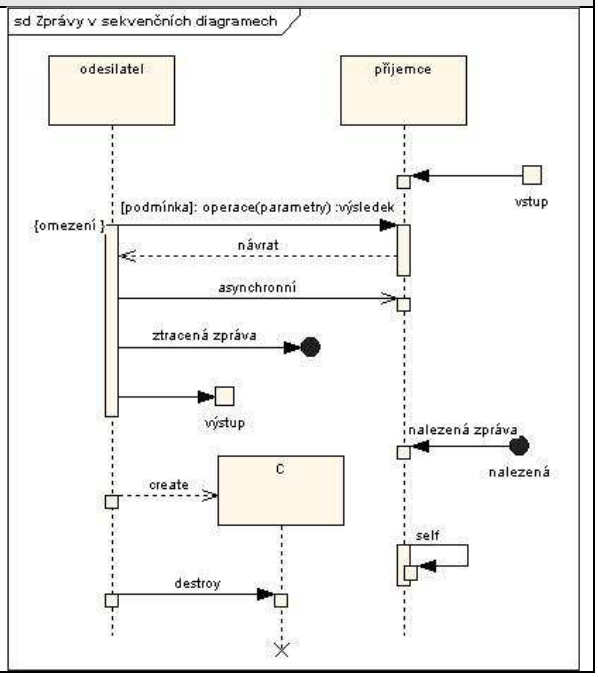
2.10 Sekvenční diagramy

Diagramy aktivity a stavové diagramy popisují synchronní a asynchronní činnosti. Pokud si systém představíme jako sadu komunikujících objektů, které uskutečňují naše požadavky, můžeme použít jako model sekvenční digramy (scénáře), nebo diagramy komunikace objektů. To jsou alternativní vyjádření komunikace mezi objekty. Elementy sekvenčních diagramů popisuje tabulka Tab. 8.

Element (prvek)	UML sémantika	UML syntaxe
objekt, instance, aktér (lifeline)	dokumentuje instance objektů, které participují při interakci	
zpráva (message)	objekty si posílají zprávy, zpravidla jako žádosti o provedení nějaké akce	
stav	dokumentuje stav objektu během scénáře	
fragment	dokumentuje různé alternativní možnosti, umožňuje strukturovaní scénářů	

Tab. 8: Elementy sekvenčních diagramů

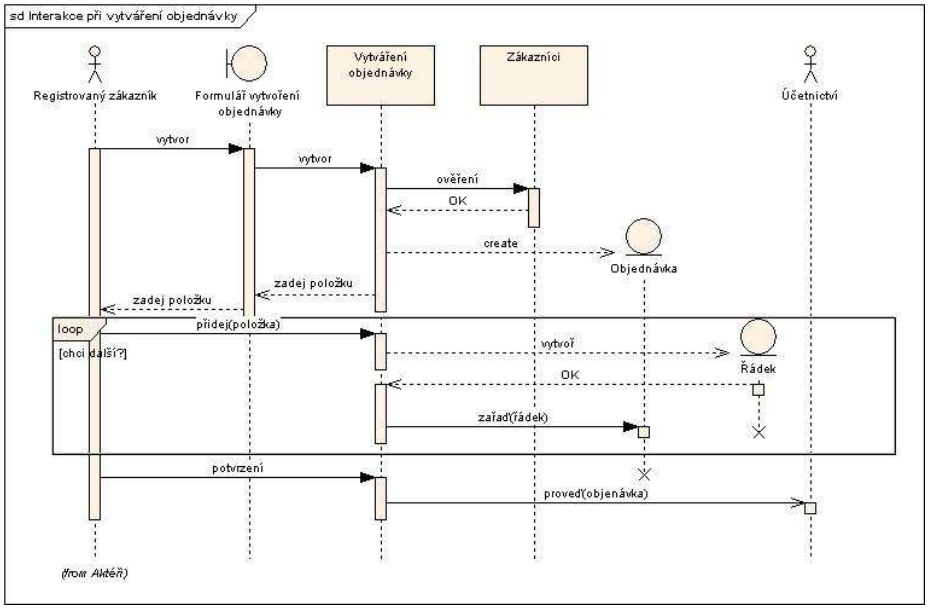
Zprávy, které si objekty posílají, mohou být synchronní, či asynchronní. Podrobnosti popisuje tabulka Tab. 9.

Element (prvek)	UML sémantika	UML syntaxe
synchronní zpráva	odesilatel čeká na odpověď	
asynchronní zpráva	odesilatel nečeká na odpověď	
návrat	návrat ze synchronního volání	
konstrukce a destrukce objektu	odesilatel vytváří, příp. ruší objekt (destrukce)	
nalezená zpráva	zpráva přichází z prostředí mimo rámec této interakce	
ztracená zpráva	zpráva nedosáhla adresáta	

Tab. 9: Zprávy v sekvenčních diagramech

Scénář interakce mezi objekty při vytváření objednávky

Jak bude probíhat interakce mezi objekty při průběhu služby „Vytvoření objednávky“. Interakci vybudí aktér „registrovaný zákazník“ – kterákoliv instance tohoto typu.



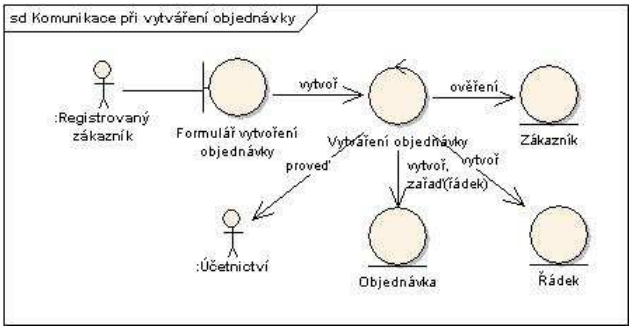
Obr. 11: Interakce při vytváření objednávky

2.11 Diagramy komunikace

Sekvenční diagramy (scénáře) popisují modelovaný systém jako sadu komunikujících objektů, které si posílají při uskutečňování požadavků zprávy. Důraz je kladen na časový aspekt návaznosti zpráv. Pro návrh může být užitečnější pohled, kdy nás více zajímá požadované rozhranní, než čas. Takovou alternativou ke scénářům jsou diagramy komunikace objektů. Elementy diagramů komunikace popisuje tabulka

Komunikace mezi objekty při vytváření objednávky

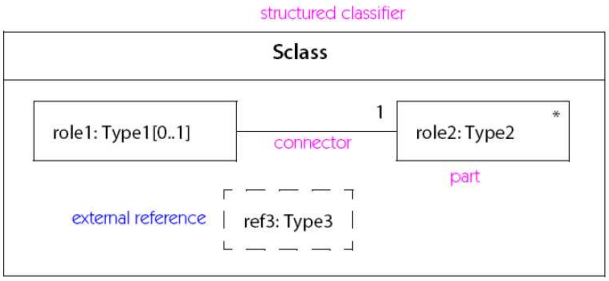
Komunikaci mezi objekty při vytváření objednávky popisuje obrázek Obr. 12. Na rozdíl od sekvenčního diagramu zdůrazňuje požadované operace.



Obr. 12: Komunikace při vytváření objednávky

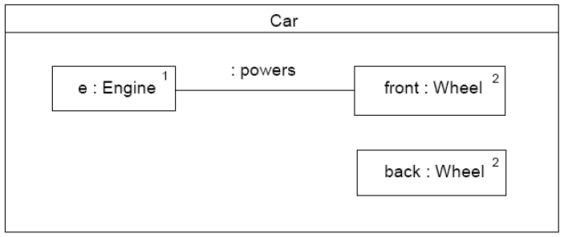
2.12 Diagramy kompositní struktury

Popis požadované struktury pomocí diagramů tříd má svá úskalí. Snadno popíšeme, že auto obsahuje jeden motor, dvě přední a dvě zadní kola. Už obtížněji se popisuje, že motor pohání přední kola auta, ve kterém je sám a nepohání nic jiného. Tyto problémy by měly řešit diagramy kompositní struktury. Jejich notaci popisuje obrázek Obr. 13.



Obr. 13: Notace diagramů kompositní struktury

Uvažme jako příklad popis struktury auta pomocí diagramu kompositní struktury



Obr. 14: Příklad diagramu kompositní struktury

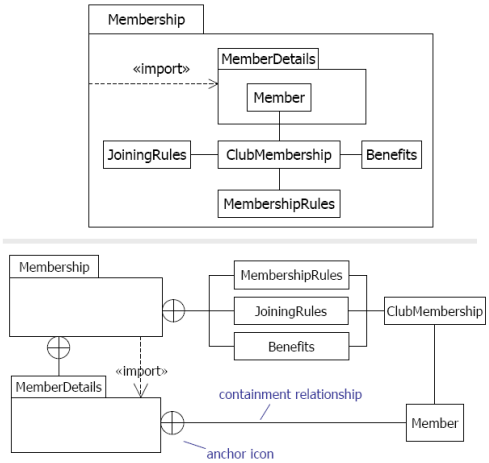
Na rozdíl od diagramu tříd se zde pracuje s instancemi. Diagram říká, že v každém autě je jeden motor, dvě přední a dvě zadní kola. Motor pohání přední kola a nepohání nic jiného.

2.13 Diagramy balíků

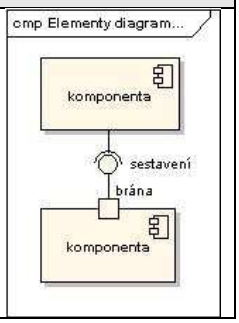
Známé psychologické pravidlo 7±2 říká, že jeden diagram by neměl obsahovat více než cca 7 (±2) prvků. Popisujeme-li komplikovanější systém, musíme popis nějakým způsobem strukturovat. UML nám pro tento účel poskytuje různé možnosti, jednou z nich jsou diagramy struktury balíků. Používají se v nich zejména ikony pro balíky, spojované různými vztahy. Notace je uvedena na obrázku Obr. 15.

2.14 Diagramy komponent

Architektura systému obecně popisuje sadu komponent, ze kterých se celek skládá a jejich vazby. Komponenty potřebné pro běh aplikace jsme popisovali diagramem nasazení. Komponenty uváděné v těchto diagramech jsou „běhové“ – jsou to části celku nasazené na výpočetní prostředky. Takové komponenty je ovšem předtím nutno nějak vytvořit, sestavit. K popisu takového sestavení lze využít diagramy komponent. Elementy diagramů komponent popisuje tabulka Tab. 10.



Obr. 15: Notace diagramů balíků

Element (prvek)	UML sémantika	UML syntaxe
komponenta	modulární část systému zapouzdřující svůj obsah, jejíž projev je nahraditelný v daném prostředí	
sestavení (assembly)	dokumentuje sestavení propojených komponent, skládá se z ikony pro poskytované rozhranní (plug, ball) a ikony pro požadované rozhranní (socket)	
Brána (port)	bod interakce mezi klasifikátorem a okolím, používá se, pokud chceme zdůraznit a pojmenovat bod interakce, příp. vyznačit, kam se v komponentě tato interakce deleguje	

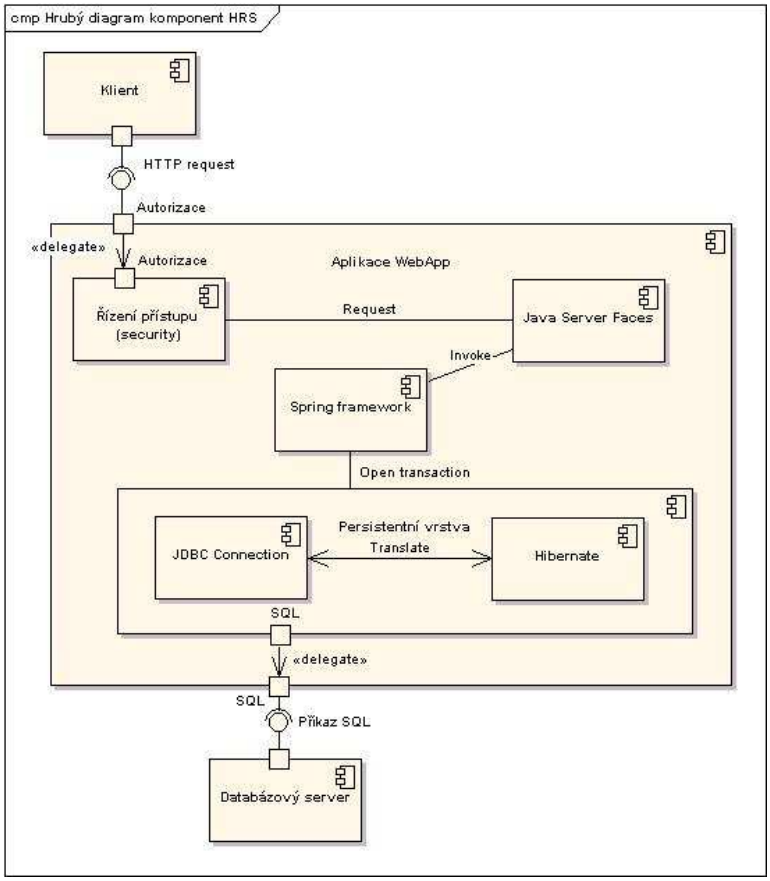
Tab. 10: Elementy diagramů komponent

Hrubý diagram komponent HRS

Systém HRS bude implementován pomocí nástroje Java Server Pages, persistentní vrstvu pro data zajistí prostředí Hibernate připojené přes JDBC k databázovému serveru. Tuto úvahu lze znázornit hrubým diagramem komponent na obrázku Obr. 16. Diagram komponent popisuje hrubou strukturu aplikace WebApp, lze rozpracovat různě podrobné verze. Mějte ovšem na mysli, že platí základní pravidlo modelování:

- *„Vytváření modelu musí mít nějaký smysl, modely vytvářené jen pro modelování samo, nejsou zpravidla k ničemu.“*

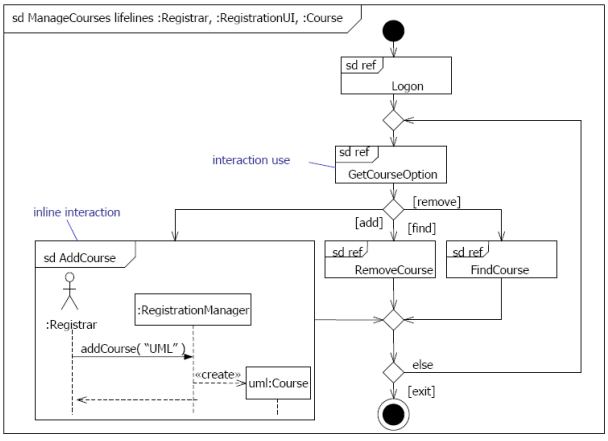
Pozn. Parafráze na Einsteinův výrok – *„Dělejte věci tak jednoduché, jak to jde, ale ne jednodušší.“*



Obr. 16: Hrubý diagram komponent HRS

2.15 Přehledové diagramy interakce

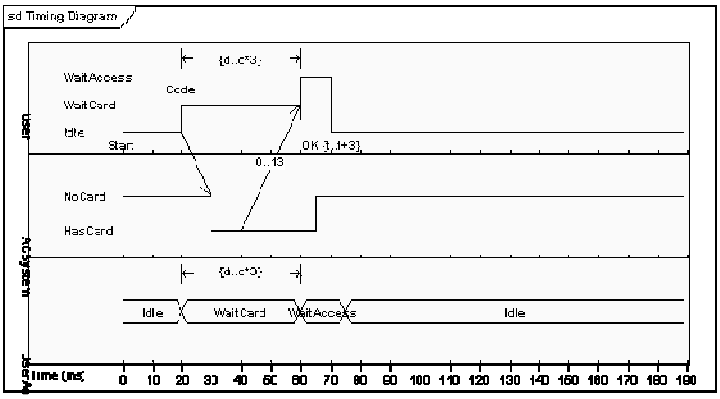
Slouží k popisu složitějších interakcí, jedná se jakýsi zobecněný diagram aktivity.



Obr. 17: Příklad přehledového diagramu interakce

2.16 Časové diagramy

Poslední typ diagramu popisuje přesněji časové závislosti. Má dvě formy, jak ukazuje obrázek Obr. 18.



Obr. 18: Notace časových diagramů

2.17 OCL (Object Constraint Language)

Součástí specifikace UML je i definice jazyka OCL, který slouží pro vyjádření artefaktů, které nelze vyjádřit obrázkem. Také jej lze využívat pro navigaci v diagramech. Zápisy v OCL mají tvar:

context <jméno> [inv|pre|post]: <výraz>

kde klíčové slovo **context** slouží pro definici kontextu pro výraz v OCL, např. zápis:

context Student inv : <výraz>

označuje, že uvedený výraz se vztahuje k instanci kontextu (třídy) Student (na kterou se lze odkazovat klíčovým slovem **self.** Klíčová slova **inv**, **pre** a **post** zastupují stereotypy <<invariant>>, <<precondition>>, <<postcondition>>. Jako příklad použijme popis omezení (zde „business rules“) pro diagram tříd z obrázku Obr. 7.

Integritní omezení v modelu HRS

V datovém modelu naší aplikace HRS musí např. platit pravidlo: *„Počáteční datum pobytu musí být menší než koncové.“* V OCL to zapíšeme:

context Pobyt inv : self.od <= self.do

Jiné pravidlo říká: *„Dvě rezervace téhož pokoje se nesmí překrývat.“*

```
context Rezervace inv : forall(Rezervace a,b |  -- pro libovolné dvě rezervace a,b
    let x = a->Pobyt in let y = b->Pobyt in     -- nechť x a y jsou rezervované pobyty
    x->Pokoj.cislo = y->Pokoj.cislo             -- které se týkají stejného pokoje
    implies y.od > x.do or y.do < x.od)         -- se nepřekrývají v čase
```

3 UML v praxi

V této části tutoriálu se podíváme na UML z pohledu toho, kdo realizuje SW projekty (nebo vývoji software blízké projekty) u zákazníků, tedy toho, kdo UML používá *v praxi*. Kapitola je členěna takto:

- S KÝM je potřeba se o tom bavit: UML je koneckonců jazyk pro sdělování informací při vývoji SW (nebo blízkých disciplin). Tohoto procesu se zúčastňují odborníci z různých oblastí a je nutné respektovat jejich zvyklosti a ochotu je změnit.
- CO máme k dispozici: tyto odstavce se zabývají jednotlivými UML diagramy a jejich konkrétním použitím.
- KDE se UML dá použít: popis jednotlivých oblastí resp. fází projektu, kde je vhodné použít UML.
- JAK zapsat modely: tato část se zabývá některými postřehy z použití nástrojů pro zaznamenání modelů v notaci UML, tzv. *CASE* nástroje.
- Něco MIMO UML: jsou určité oblasti v rámci procesu tvorby SW, pro které UML nemá podporu nebo někdy ani není vhodné je použít. Vytyčení hranic, kde UML ano a kde ne, může ukázat jeho použitelnost v ostřejších barvách.

Text vychází ze zkušeností, postřehů a chyb autora na konkrétních projektech z problémových oblasti jako státní správa, telekomunikace a bankovnictví.

Zkušenosti byly získány těmito způsoby:

- learning by doing, tzn. účastí na projektech.
- studiem mnoha postupů a doporučení v dnes již velmi rozsáhlé literatuře, zejména však na webu. V závěru tohoto textu jakož i v rámci tutoriálu bude uveden seznam doporučených studijních materiálů.

3.1 S KÝM komunikujeme?

V této části tutoriálu rozdělíme modely a diagramy, které UML poskytuje, podle jejich zaměření na určitou skupinu účastníků vývojového procesu. Jak je vidět s části tutoriálu UML v teorii, UML poskytuje velkou řadu silných analytických prostředků. V této části tutoriálu se podíváme hlavně na to, komu jsou tyto prostředky určeny a jak přispívají ke komunikaci jednotlivých lidí (rolí) na projektu.

Část výkladu bude věnována diskuzi a pokusům definovat jen určitou podmnožinu jazyka UML. Budou uvedeny odkazy na příslušné informační zdroje.

3.2 CO máme k dispozici?

V této části tutoriálu probereme zkušenosti získané při vytváření některých modelů a diagramů UML na konkrétních projektech.

SW aplikace musí být analyzována z těchto hledisek:

- Informace, které jsou aplikací spravovány (statický pohled).
- Chování, které poskytuje svým uživatelům (dynamický pohled).
- Uživatelské rozhraní, které uživatelé používají pro ovládání aplikace.
- Architektura, která definuje jak je aplikace sestavena.

Pro zachycení téměř všech těchto hledisek poskytuje UML modely a diagramy. Jednotlivým modelům a diagramům jsou věnovány zvláštní odstavce.

Modely, pohledy, diagramy, balíčky:

Zaměříme se na ty prostředky UML, které slouží k lepšímu uspořádání modelu, zvýšení přehlednosti a udržovatelnosti modelu.
Podle definice [7] lze analýzu členit na modely, pohledy (view), diagramy a balíky/balíčky (package). Package diagram neslouží k vyjádření žádného specifického aspektu vyvíjeného systému, ale je velmi vhodný pro zpřehlednění modelu, zejména k rozdělení složitějších modelu na menší logické celky.

Elementy modelů jsou rozděleny do diagramů. Diagramy sice vydají za 1000 slov, ale jen v případě, že představují logicky konzistentní skupinu elementů a jsou dostatečně přehledné – dalo by se říct, že mají určitou „estetickou“ hodnotu. Další diskuze k této problematice bude uvedena v části JAK.

Class/objektový diagram

Class diagram slouží pro popis struktur nebo výskytů dat, se kterými aplikace pracuje. Class diagram je možné použít na různých úrovních abstrakce buď podle zaměření projektu, nebo v závislosti na etapě, ve které je použit. Ukázky použití class diagramu z různých *perspektiv* (pojem zavedený v [8]). Perspektivy jsou rozlišeny na konceptuální, esenciální a implementační.

Objektový diagram nebývá častou součástí analýz, ale pro lepší pochopení abstraktnějších class modelů je cennou pomůckou. V rámci tutoriálu budou ukázány příklady použití objektového diagramu.

Use Case model

Use Case model (model případů užití) se použije pro definici toho, jakou SW podporu dostane budoucí uživatel při jeho práci (často se hodí říct při jeho *businessu*, neboť se tím naváže na business model příslušné domény). Tento model je základním prostředkem pro komunikaci analytika s uživatelem. Use case model neobsahuje jen diagramy, ale i textové části. Textové popisy musí mít jasně definovanou strukturu (tzv. *scénáře*) a musí být pro ně dohodnuto, jakých hledisek specifikace aplikace se mají týkat. V tutoriálu bude uvedeno několik doporučení jak psát scénáře a do jaké míry se v těchto scénářích zabývat i jinými hledisky vyvíjeného systému, jako je například grafické uživatelské rozhraní (anglická zkratka - GUI).

Use Case model, stejně jako většina modelů v UML, prošel od jeho zavedení do prvních verzí UML, značným vývojem - je teď složitější. S ohledem na jeho zaměření na uživatele je to však i kontraproduktivní, jak se ukazuje v doporučeních renomovaných autorů (viz. [9], [10]), kteří varují před (hojným) používáním některých prvků modelu. Příklady jsou stereotypy asociací <<includes>>, <<extends>> a <<generalize>>.

V tutoriálu ukážeme příklady Use Case diagramů a jejich textových popisů.

Aktivity diagram

Aktivity diagramy (AD) jsou někdy označovány jako control flow digramy přenesené do prostředí objektové analýzy a návrhu. Aktivity diagramy jsou zjevně nejsložitější ve smyslu počtu elementů, jakož i ve smyslu sémantiky jednotlivých elementů. Došlo u nich k nejrozsáhlejším změnám od verze 1.5. Použití activity diagramů je možná poněkud složitější pro analytiky zvyklé specifikovat chování aplikace pomocí DFD. Tutoriál se bude zabývat těmito tématy:

- Aktivity diagramy mají přesně definovanou sémantiku pomocí tzv. coloured Petri-nets, kdy je pro každý element definováno jakým způsobem přijímá/vysílá tzv. *tokeny*.
- Bude proveden výklad rozdílu sémantiky dataflow a control flow, některé řídicí konstrukce, upozorníme na webové stránky s animovanou sémantikou řídicích elementů.
- Budou ukázány rozdíly AD a DFD (push a pull metoda)
- Krátká diskuze bude věnována chybám a omylům, kterých se autor při používání AD dopustil.

Sekvenční diagram

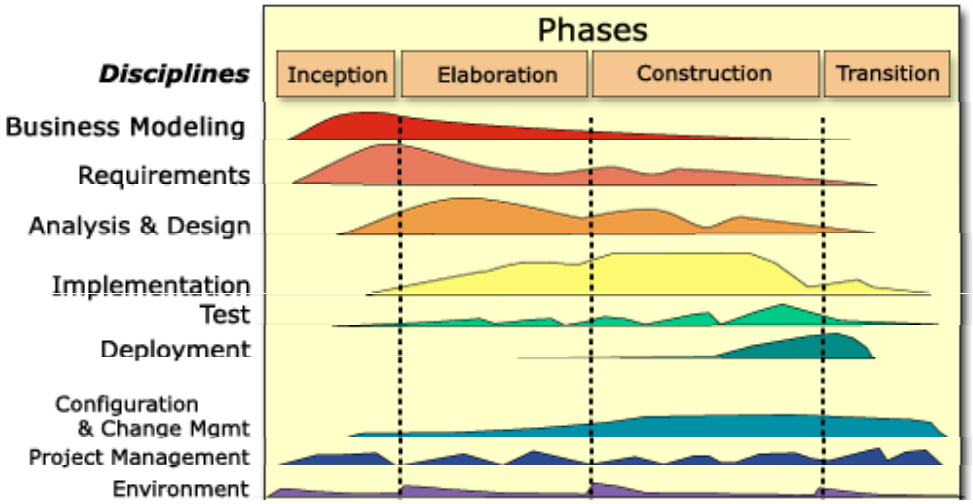
Sekvenční diagram se používá pro znázornění komunikace mezi několika objekty na základě zasílání zpráv. Většinou je chápán jako jeden z důležitých prostředků v rámci návrhu aplikace, který propojuje strukturální a dynamickou část analýzy. V tutoriálu ukážeme příklady použití, některé i z oblasti business analýzy, kde jde o vyjádření komunikace vyvíjené aplikace s okolními systémy.

Stavový diagram

Stavový diagram je nástrojem pro vyjádření životního cyklu jednoho objektu. Tento diagram lze použít na různých úrovních abstrakce a v různých typech projektů. Určitá potíž je v tom, že není uživateli příliš oblíben, což snižuje jeho použitelnost. Vyjadřovací schopnosti tohoto diagramu byly v UML 2 značně rozšířeny, což výrazně zlepšuje jeho schopnost specifikovat chování aplikace.

3.3 KDE se UML používá?

Pokud se chceme bavit o oblastech použití UML, nelze to udělat bez toho, že bychom jednotlivé modely a diagramy, tedy to CO máme k dispozici, nevztáhli k tomu KDE je vhodné tyto prostředky použít. Dlouhodobým souputníkem UML v oblasti metodik vedení projektů je tzv. *Unified Process (UP)*, který definuje základní fáze vývoje SW. Základní oblasti použití, na následujícím obrázku nazvané disciplíny, probereme pak podrobněji.



Obr. 19: Disciplíny a fáze UP

V tutoriálu budou probrány jednotlivé modely a jejich vztah k disciplínám (etapám) UP. Výklad bude doplněn příklady z konkrétních projektů.

3.4 JAK zapsat modely?

Zaznamenání modelů UML se provádí pomocí nástrojů typu CASE. Prakticky všechny tyto SW nástroje poskytují mnoho možností, jak jednotlivé elementy modelu popsat. Analytik tedy nepotřebuje znát jenom jazyk UML, ale také podstatné rysy nástroje CASE, použitého na projektu.

V této části tutoriálu se budeme zabývat tématy, která „se točí“ kolem efektivního použití CASE nástrojů, zejména v situaci většího analytického týmu. Tak jako u samotného jazyka UML, je i v případě CASE nástrojů vhodné zahajovat projekt s jasnými dohodami, jak budou jednotlivé modely, diagramy a elementy dokumentovány.

CASE nástroje poskytují řadu funkcí, které jsou tzv. *mimo* UML. Těmto otázkám věnujeme samostatnou část. Zde se jen zmíníme o tématech jako jsou: reportování, údržba vztahů mezi modely a jejich částmi, křížové matice, metriky projektů a podobné rysy, bez kterých se však praktické používání CASE nástrojů neobejde. Tutoriál bude doprovázen krátkou ukázkou práce v CASE nástroji Enterprise Architect (EA).

3.5 Je něco MIMO UML?

V této části tutoriálu se budeme zabývat některými otázkami, které jsou tzv. *mimo* UML. Jsou to témata, která nejsou ve specifikaci UML uvedena a přitom do SW projektů patří a patří tím i do základní výbavy každého analytika.

Mimo UML jsou z tohoto pohledu tato témata:

- požadavky (requirements): každý větší projekt začíná sběrem požadavků. UML nemá odpovídající model pro zaznamenání požadavků.
- specifikace GUI: u většiny projektů se zájem budoucích uživatelů soustřeďuje na otázku „Jak se to bude ovládat?“. UML nemá odpovídající prostředky pro popis GUI. S problematikou GUI se pojí i otázka jak ho popisovat v jiných modelech (například v Use Case modelu).
- specifikace business pravidel: UML sice obsahuje speciální jazyk OCL pro zaznamenání business pravidel, ale ten je bohužel odmítán i těmi, které lze považovat za fandy UML.
- používání textu: jak jsme uvedli již dříve, UML nejsou jen diagramy. Bez srozumitelného textového popisu je řada diagramů nesrozumitelná. Použití textu může často zpřístupnit UML modely různým uživatelům (od manažerů až po koncové uživatele), může někdy dramaticky zkrátit jinak velmi formální zápisy.

Řadu těchto rysů podporují právě CASE nástroje.

4 Závěr

Tutoriál se zabývá problematikou jazyka UML jak z teoretického tak praktického hlediska. Zvládnutí těchto prostředků vyžaduje otevřenou hlavu a ochotu studovat řadu odborných textů a doporučení renomovaných autorů. A to všechno v situaci, kdy dále budou do UML přibývat nové a nové prvky, diagramy a modely.

Na závěr si tedy můžeme dovolit dát jednu radu:

„Používejte UML a máte co analyzovat i bez projektů!“

Literatura

1. Sémantika diagramů aktivity s animací: http://www.workflowpatterns.com/
2. Weilkiens, T., Oestereich, B.: UML2 Certification Guide. Elsevier, Inc., 2007.
3. Arlow, J., Neustadt, I.: UML a unifikovaný proces vývoje aplikací. Computer Press, Brno, 2007.
4. ISO/IEC 19501:2005(E): Unified Modeling Language Specification. Version 1.4.2, OMG formal/05-04-01. January 2005
5. Richta, K., Svačina, J.: UML – teorie a praxe. In: DATAKON 2002. str.1-26, ISBN 80-210-2958-7, Masarykova Universita, Brno, 2002.
6. Štourač, D.: Modelovací jazyk UML. In: DATASEM 1999. CZ-COMPEX, 1999.
7. OMG: Unified Modeling Language. Version 2.1.2, OMG formal/07-11-04, November 2007.
8. S.Cook, J.Daniels: DESIGNING OBJECT SYSTEMS, Object – Oriented Modelling with Syntropy, Prentice Hall, 1994
9. K.Bittner, I.Spence: Use Case Modeling, Addison-Wesley, 2006
10. M.Fowler, K.Scott: UML Destiled, Third Edition, Addison Wesley, Object Technology series2000

Annotation:
This paper deals with the theoretical and practical aspects of the Unified Modeling Language UML.