SLEZSKÁ UNIVERZITA
OBCHODNĚ PODNIKATELSKÁ FAKULTA V KARVINÉ

Objektové metody modelování v příkladech

Distanční studijní text

Tomáš Barčák, Zdeněk Franěk

Karviná 2024

Obor: Informační a komunikační technologie (ICT), Vývoj a analýzy softwaru a aplikací

Klíčová slova: Analýza, software, diagramy UML, metodika RUP, objekty, třídy, polymorfismus, dědičnost.

Anotace: Tento učební text se zabývá analýzou návrhu software s využitím Unified Modeling Language (UML) diagramů a metodiky Rational Unified Process (RUP). Je určen pro studenty 1. ročníku studijního programu Manažerská informatika v navazujícím stupni studia, zejména v kombinované formě výuky. Z tohoto důvodu je forma učebního textu koncipována tak, aby studenti měli k dispozici text postačující k ovládnutí dané problematiky a zároveň si mohli své znalosti po každé kapitole ověřit pomocí testových otázek nebo úkolů.

Hlavním tématem studijní opory je popis analytických postupů při návrhu informačních systémů a vývoje software s využitím jazyka UML a metodiky RUP. Modelovací jazyk UML je základní nástroj pro objektovou analýzu při návrhu software a informačních systémů. Tato studijní opora je určena zejména pro studenty v distanční formě studia a tomu odpovídá jiná struktura studijní opory dle předepsané wordovské šablony. V této studijní distanční opoře je kladen důraz na praktické předvedení možností objektových technik modelování, a proto je výklad provázen řadou praktických příkladů. Po každé kapitole jsou zařazeny testové otázky tak, aby si student osvojené poznatky mohl sám otestovat.

Studijní opora je dostupná ve studijních materiálech informačního systému Slezské univerzity v Opavě povinně volitelného předmětu „Objektové metody modelování“ magisterského studijního programu „Manažerská informatika“.

Objektově orientované modelovací techniky a jazyk UML jsou ústředním tématem této opory.

Ve tomto vydání studijní opory byla upravena úvodní kapitola, přepracovány a doplněny návody pro využití software v 8. kapitole. Dále pak devátá kapitola byl rozšířena tak, aby text byl komplexnější a dával větší smysl a přehled o metodice RUP. Zcela byla přepracována koncepce 10. kapitoly. Příklady modelování byly přepracovány a doplněny případovou studií a ukázkovými diagramy. Rovněž byl doplněn a upraven seznam literatury.

Autor: **Ing. Tomáš Barčák, Ph.D.**
RNDr. Zdeněk Franěk, Ph.D.

Obsah

ÚVODEM

Návrh informačních systémů (dále jen IS), metodika jejich vytváření, programování systémů a aplikací, vytváření software, objektové metody modelování s využitím jazyka Unified Modeling Language (dále jen UML) jsou na vrcholu témat tvůrčí práce v oblasti informačních technologií (dále jen IT) technologií.

Učební text, který je z výše uvedených oblastí návrhu informačních systému zaměřen především na objektové metody modelování s využitím jazyka UML a metodiku vytváření software, je určen studentům 1. ročníků studijního programu Manažerská informatika na Obchodně podnikatelské fakultě v Karviné, Slezské univerzity v Opavě (dále jen SU OPF). Je zaměřen na kombinovanou formu studia a tomu je určena forma textu. Text je provázen ikonami, zdůrazňujícími hlavní rysy IT technik z výše uvedenou tématikou. Na začátku každé kapitoly je shrnutí, čemu se studenti naučí, následuje výklad a příklad, pokud to umožňuje charakter tématu. Vždy jsou graficky zdůrazněny hlavní myšlenky, resp. poznatky. Na konci každé kapitoly jsou uvedeny testové otázky k procvičení probrané látky. Správné odpovědi pro samokontrolu jsou přiloženy hned za otázkami.

Pochopení textu této distanční studijní opory předpokládá u studentů základní znalosti z oblasti informačních technologií. Je určen studentům navazujícímu stupni studia, především v kombinované formě studia. Tyto předpokládané znalosti odpovídají znalostem, získaných absolvováním bakalářského stupně studia studijního programu Manažerská informatika na OPF. Jedná se zejména o zvládnutí základních poznatků z předmětu databázové systémy, algoritmy a datové a procesní modelování. Všechna tato témata jsou součástí studia v bakalářském stupni, studijního programu Manažerská informatika. Přesto lze k textu přistoupit bez předchozího studia a v případě neporozumění textu si znalosti doplnit z příslušných částí příslušných studijních opor.

Text je doprovázen případovou studií a látka je ilustrovaná příklady, na kterých lze porozumět předchozí teorii. Doporučený postup při práci s textem je následující: Přečíst si důkladně teorii a hned si ji ověřit na příkladu. Poté se znovu vrátit k teorii a pomocí příkladu objasnit sporná nebo těžká místa. Zpětným pročítáním textu lze tzv. iterační metodou přírůstků znalostí dospět k pochopení probíraných témat.

Zvláštní pozornost doporučujeme věnovat rovněž metodice návrhu informačních systémů. Je nutno zdůraznit, že bez dobré metodiky nelze dosáhnout úspěchu při návrhu IS. Metodika Unified Process (dále jen UP), objektové metody modelování a UML diagramy spolu úzce souvisí.

Je třeba zdůraznit, že tento učební text se věnuje výhradně fázi analýzy před zahájením programování informačního systému, resp. software obecně. Cílem autora bylo popsat všechny techniky tak, aby po jejich aplikaci bylo možno zahájit programátorské práce a aby si zúčastněné strany (zadavatel, uživatel, analytik a programátor) plně rozuměli.

Samozřejmě to nevylučuje to, že se UML diagramy často používají i pro nasazení a dokumentaci programových systémů.

Tato studijní opora je součástí e-learningového kurzu na http://elearning.opf.slu.cz, který je dostupný pro studenty předmětu „Objektové metody modelování“.

Text byl sestaven na základě využití literatury, zkušeností autora a na základě přednášek a seminářů během výuky předmětu.

V práci jsou využity poznatky při zpracování seminárních prací studenty ve spolupráci s lektorem předmětu.

V předmětu je využíván software, při jednoduchých úlohách je to MS VISIO a šablony pro UML. Pro složitější úlohy byly v předmětu využívány CASE nástroje. Nejdříve to byl CASE firmy IBM Rational IBM Architect. Později velmi kvalitní a nejvíce rozšířený CASE nástroj firmy SPARX Enterprise Architect. Tento software se při rozvoji předmětu plánuje využívat jako číslo 1.

Poděkování za spolupráci na této učební opoře patří samozřejmě autorům literatury, studentům-účastníkům přednášek a seminářů, dále pak spolupracovníkům z katedry matematiky a informatiky SU OPF a za podporu našim blízkým.

RYCHLÝ NÁHLED STUDIJNÍ OPORY

Cílem studijní opory je srozumitelnou formou s využitím ikon a struktury používané pro kombinovanou formu výuky seznámit studenty s teorií objektového modelování systémů a jejím významem pro projektování informačních systémů. Předmět seznámí studenty s historickým vývojem objektového přístupu a používanými standardy v dané oblasti. V textu jsou studenti seznámeni s metodikou RUP (Rational Unified Process). Hlavní náplň opory je věnována jazyku UML (Unified Modeling Language). Jazyk UML byl koncipován pro návrh, analýzu, tvorbu a dokumentaci informačních systémů s objektově orientovaným přístupem. V rámci studia učební opory studenti získají znalosti standardního způsobu zápisu – koncepce návrhu systému, jako jsou business procesy a systémové funkce. Dále se naučí tvořit konkrétní prvky, jako jsou například znovupoužitelné programové komponenty a databázová schémata. Na příkladech je demonstrována praktická práce s jazykem UML. V textu jsou procvičeny základní postupy při vývoji software s využitím Enterprise Architect (označuje se zkratkou EA) firmy SPARX, který patří do rodiny Computer Aided Software Engineering (ve zkratce CASE) a šablony pro UML diagramy produktu VISIO firmy Microsoft. Studenti jsou seznámeni se softwarem IBM Rational Enterprise Architect, průkopníkem mezi CASE softwarovými nástroji na poli vývoje informačních systémů.

Učební text se dělí do těchto témat:

1. Úvod do objektového modelování
2. Shrnutí základních pojmů objektově orientované analýzy a návrhu software. Pojem objekt. Základní koncepty: abstrakce, zapouzdření, skrývání informací, třídy, dědičnost, interface.
3. Popis jazyka UML
4. Co je UML, objekty a jazyk UML, struktura jazyka, stavební bloky, vyjádření tříd, atributů a operací. Obecný přehled diagramů UML. Diagramy UML: případy užití, stavové diagramy, diagramy sekvencí, diagramy spolupráce, diagramy tříd, diagramy činností, diagramy architektury.
5. Software produkty pro práci v UML
6. Přehled softwarových nástrojů pro objektové modelování. Představení a zpřístupnění software EA SPARX a MS VISIO - část pro kreslení UML diagramů. Případy užití.
7. Metodika Unified Process (UP) a Rational Unified Process (RUP)
8. Hlavní principy moderního iterativního vývoje softwaru metodikou RUP.
9. Jednotlivé fáze životního cyklu projektu, který vyvíjí software - účel, možnosti, rizika vývoje.
10. Případové studie návrhu informačního systému s využitím UML a metodiky RUP.

1 ÚVOD DO OBJEKTOVĚ ORIENTOVANÉHO MODELOVÁNÍ

RYCHLÝ NÁHLED KAPITOLY

Úvodní kapitola učebního textu seznamuje studenty s principy objektově orientovaného návrhu software. Objektově orientované modelování ve zkratce OOM pochází z anglického Object-Oriented Modeling. V první kapitole je popsána historie vzniku objektově orientovaného přístupu vývoje software a srovnání s předchozím procedurálním přístupem. V kapitole jsou popsány základní pojmy jako je třída, objekt, zapouzdření, dědičnost a polymorfismus a některé další vlastnosti OOM. Objektově orientované modelování pokrývá analýzu, programování a nasazení software.

CÍLE KAPITOLY

Cílem úvodní kapitoly je seznámit čtenáře se základními pojmy objektově orientovaného modelování při vytváření software, obecněji informačních systémů. Základní pojmy popsat a vysvětlit, a také doložit na příkladech.

ČAS POTŘEBNÝ KE STUDIU

100 minut.

KLÍČOVÁ SLOVA KAPITOLY

Objekt, třída, polymorfismus, dědičnost, zapouzdření, překrývání metod.

1.1 Od strukturálního pojetí k objektovému

Strukturální pojetí programování

Strukturální pojetí programování je charakteristické tím, že naprogramovaný kód pracuje přímo s daty. Vývoj programového kódu aplikace byl sestaven strukturovaně. Hlavním rysem analýzy při vývoji aplikace bylo to, že se aplikace dělila na dynamickou funkční část a statickou datovou část. Data byla ukládána v jednotlivých souborech nebo později v relačních databázích. Programový kód byl psán shora dolů s využitím volání funkcí. To na jedné straně zpřehledňovalo programování a na straně druhé zavádělo prvky opakované použitelnosti, tzv. re-use. Při tomto přístupu k programování narůstala složitost programů a používané analytické postupy při návrhu softwaru narážely na velké problémy propojení vrstvy datové a funkční. Výpočty a zacházení se stejnými daty se prováděla na mnoha místech programového kódu a bylo hodně komplikované provádět změny a další přidávání funkcionalit systému. Po zavedení architektury klient - server nastaly další potíže, jak správně vyvíjet informační systémy.

Z výše uvedených důvodů se v programátorské komunitě ve světě začal používat objektově orientovaný přístup. Jedním z hlavních cílů objektově orientované analýzy, návrhu a programování je další zvýšení produktivity vývojářských prací. [Kan2004]

Objektově orientovaný přístup vývoje software

Objektově orientovaný přístup vývoje software je mladší technika navazující na strukturovaný přístup. Tento přístup je založen na objektech. Objekty jsou struktury, které mají definované vlastnosti (atributy) a své chování (operace), které daný projekt může provádět. Informační systém (IS), resp. software takto vyvinutý je chápán jako množina spolupracujících objektů. Tento přístup umožňují programovací jazyky jako je Java, C#, Smaltalk, atd. Návrh – analýza při tvorbě informačních systémů je reprezentována CASE nástroji (Computer Aided Software Engineering) a Unified Modeling Language. Modelovací jazyk UML je naprosto v souladu s objektovým přístupem. V systémech napsaných pomocí OOM se daleko více uplatňuje znovu-použitelnost. Zavádí se pojmy třída, dědičnost, zapouzdření, komponenty, distribuované objekty a jiné, které oproti strukturálnímu přístupu výrazně zvyšují produktivitu analýzy a programování systémů. [Kan2004]

CHARAKTERISTIKA KÓDU PROGRAMU

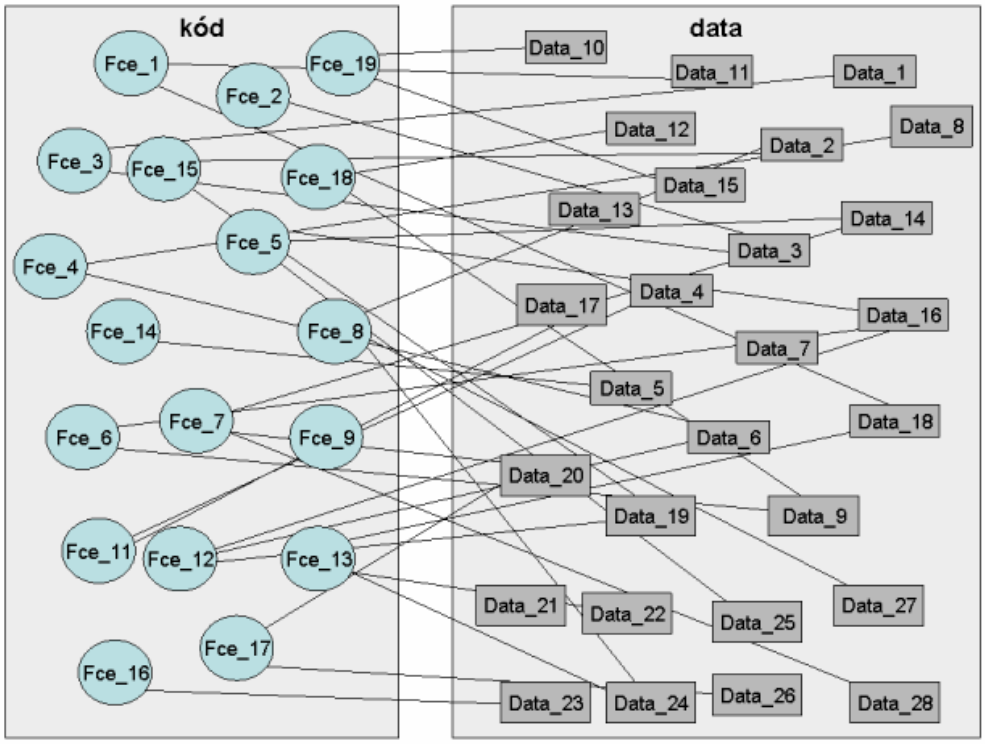
Definujme si kód: tyto prvky čtou a mění data; jsou to výkonné části informačního systému, provádějí nějakou činnost; mohou to být např. binární programy (či jejich části - moduly, funkce, procedury,...), skripty, triggery.

CHARAKTERISTIKA DAT

Definujme si data: jak proměnné (lokální či globální) držené jen v paměti (po vypnutí počítače se jejich obsah, ale vlastně i jejich samotná existence "ztratí"), tak perzistentní data (soubory, řádky databázových tabulek, apod.).

Kód pracuje s daty

Charakteristický obrázek 1 - část systému, kde kód pracuje s daty ve strukturovaném přístupu návrhu a naprogramování software:



Obrázek 1: Strukturální pojetí - kód a data, zdroj: http://mpavus.wz.cz

Tato modelová situace, kdy kód pracuje s daty, přináší různé problémy. Ukážeme si některé z těchto problémů, s jejichž řešením nám může pomoci objektový přístup.

Při vývoji IS musíme provádět transformaci požadavků do software, resp. do zdrojového kódu, který pracuje s daty. Nestačí nám prostě namodelovat realitu, ale navíc ji musíme rozdělit na oblast dat a na oblast kódu. Tato transformace nás stojí jisté úsilí, a důsledky této transformace se "nesou" celým dalším vývojem a údržbou IS.

Problémy ve strukturovaném přístupu programování

Tento přístup a postup nám ve fázi údržby a následného rozvoje může způsobit daleko větší obtíže, než které jsme si prožili při prvotním vytvoření IS.

Vznikají problémy při údržbě a dalším rozvoji IS, pokud chceme přidat/změnit funkcionalitu. Nelze často prakticky zjistit, která data jsou aktualizována, a kterou funkcí. Navíc daná funkce může vyvolávat další funkce a ty mohou spouštět další funkce, a až v těchto volaných funkcích může být pracováno s určitou oblastí dat. Toto čtení/měnění se může provádět jen za určitých podmínek, jejichž vyhodnocení může být velmi komplikované. Vzniká tak problém při údržbě a dalším rozvoji IS, jak jednoduše zajistit, aby obsah dat byl validní.

A další problémy vznikají při ladění a testování programového kódu s lokálními daty (předávanými jako parametr) a globálními daty (měnitelná i bez jejich předávání). V souvislosti s tím vzniká další otázka, jak dokonale ladit a testovat jednotlivé funkce systému, resp. jednotlivé situace v systému, které mohou nastat?

Další problémy přináší znovu-použitelnost (re-use).

K ZAPAMATOVÁNÍ

Na základě výše uvedených problémů při strukturálním přístupu byla vyvinuta IT technologie „Objektově orientovaná analýza a design“ pro tvorbu software/informačních systémů.

1.2 Objekt

DEFINICE

Objekt jako seskupení dat a funkcionality

Objekt je základní abstraktní jednotkou používanou v objektovém modelování. **Objekt je seskupením dat a funkcionality**, které jsou spolu spojeny za účelem plnění soudržné množiny zodpovědností. Objekt má svou identitu, vlastnosti, chování a zodpovědnost. [Kan2004]

Objekt je uzavřený, lokální data jsou obalena metodami, ale ani metody nejsou zvenčí přístupné - jediný způsob, jak použít objekt či jak s ním komunikovat, je poslat objektu

zprávu. Po zaslání zprávy objekt spustí svou metodu, která může použít atributy objektu a další metody objektu, může také poslat zprávu jinému objektu.

Základní pojmy charakterizující objekt byly podrobně popsány v elektronickém učebním textu, viz literatura [Fra2014], proto zde uvádíme jen jejich přehled s odkazem na elearningový kurz k předmětu Objektově orientované modelování:

Základní pojmy charakterizující objekt

- Zapouzdření
- Metody (chování)
- Atributy (lokální data)
- Odkaz na jiný objekt: pokud objekt zná odkaz na jiný objekt, může mu poslat zprávu.
- Zasílání zpráv

SAMOSTATNÝ ÚKOL

Prostudujte důkladně definici objektu a jeho základní vlastnosti a porovnejte, jakým způsobem popisují různí autoři, viz literatura tohoto učebního textu.

ÚKOL K ZAMYŠLENÍ

Dva příklady k objektovému myšlení: Objekt je černá skříňka a objekt jako HW, Komponenty, upraveno a inspirováno podle: http://mpavus.wz.cz/oo/

OBJEKT JE BLACK-BOX:

Mějme „black-box“ s několika tlačítky a kontrolkami. Může to být mobil, televize, automobil, počítač, mikrovlnka. Pro použití některého z objektů potřebujeme vědět, jaké má tento objekt chování a jaké tlačítko mám zmáčknout, aby se objekt zachoval tak, jak právě potřebujeme.

V případě mobilu: abych ho uměl zapnout, abych dovedl vytočit číslo, přijmout hovor, uložit své kontakty, abych dovedl zesílit či ztlumit hlasitost. Naprosto mě nezajímá, co je uvnitř. To znamená, že pokud požádám například zmáčknutím příslušného tlačítka o zobrazení kontaktů, mobil to provede. Mobil použije po obdržení zprávy (zmáčknutím

tlačítka) jednu nebo více metod, případně požádá o služby další objekty, které jsou připraveny, tak zvaně „zabaleny“, uvnitř v mobilu.

OBJEKT JAKO HW:

Komponenty personálních počítačů mají výše uvedené objektové vlastnosti. Jednotlivé komponenty (např. paměť, grafická karta, pevný disk, procesor ...) mají chování jako objekty:

Po správném složení a zapojení personálního počítače tyto objekty spolu spolupracují: Jeden objekt požaduje po jiném objektu, resp. jedna komponenta žádá jinou komponentu, o provedení služeb, které tento objekt poskytuje. Výsledkem spolupráce všech objektů je zajištění správné fungování systému. Pokud komponenty počítače správně spolupracují, je to funkční PC.

KOMPONENTY

Komponenty mají vnitřní paměť, vnitřní metody a lze je skládat. Komponenty jsou podrobně popsány v předchozím elektronickém skriptu [FRA2014], které je nedílnou součástí e-learningového kurzu předmětu „Objektové metody modelování“.

1.2.1 TŘÍDA

DEFINICE

Třída reprezentuje šablonu pro objekty a popisuje interní strukturu těchto objektů.

Třída objektů je definována svými atributy a metodami

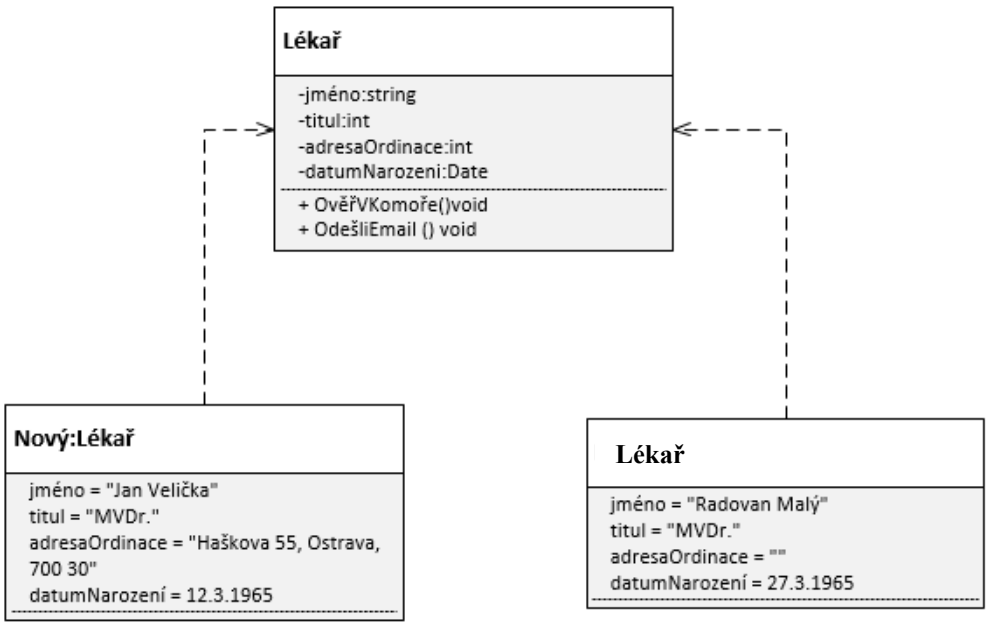
Pro návrh objektových systémů se předpokládá, že je třeba navrhnout model tříd objektů (Class model), který v podstatě nezobrazuje jednotlivé objekty, ale šablonu (předpis) pro vytvoření objektů - to je třída objektů. Třída objektů je definována svými atributy a metodami. Při návrhu třídy neuvažujeme o konkrétním naplnění atributů, pouze definujeme jejich název a typ. Při vzniku instance objektu (skutečný objekt) se atributům přiřadí skutečné hodnoty.

Diagramy tříd zobrazují statickou stránku systému

Diagramy tříd zobrazují statickou stránku systému, především vztahy mezi třídami. Vztahy, které jednotlivé třídy navzájem pojí, jsou asociace, agregace, kompozice, specializace/generalizace. Podřízenost jednoho objektu vůči druhému je v analýze chápána dvojím způsobem, buď jako agregace, nebo jako kompozice. V obou případech se však jedná o vztah dvou objektů, z nichž podřízený objekt má svou objektovou referenci vloženu do prvního objektu. K detailnímu vysvětlení jednotlivých typů vazeb se dostaneme v následujícím textu.[Kan2004]

ŘEŠENÁ ÚLOHA

Příklad: V informačním systému evidujícím lékaře máme třídu Lékař s atributy uchovávajícími informace o lékaři (jméno, adresa, atd.) a se dvěma metodami - ty umí ověřit členství lékaře v lékařské komoře a odeslat lékaři e-mail. Na následujícím obrázku 2 je znázorněna třída lékař se dvěma vytvořenými objekty nový lékař (právě zaváděný do systému) a hlavní lékař ČR.



Obrázek 2: Třída s dvěma vytvořenými objekty (instancemi třídy), zdroj vlastní

1.2.2 STRUKTURA TŘÍD

Dva principy, na kterých je třída založena

Struktura tříd je založena na dvou principech, na zodpovědnosti třídy a na zapouzdření třídy. Zopakujme si, co si pod těmito pojmy představujeme. Zodpovědnost třídy je jedním z klíčových faktorů objektově orientované analýzy a návrhu a znamená, že námi definovaný objekt nese zodpovědnost za danou problematiku (tj. „to o čem uchovává informace a chování") a žádný jiný objekt se nesmí „plést do této zodpovědnosti".

1.2.3 ZOBECNĚNÍ (GENERALIZACE-SPECIALIZACE), DĚDĚNÍ (INHERITANCE)

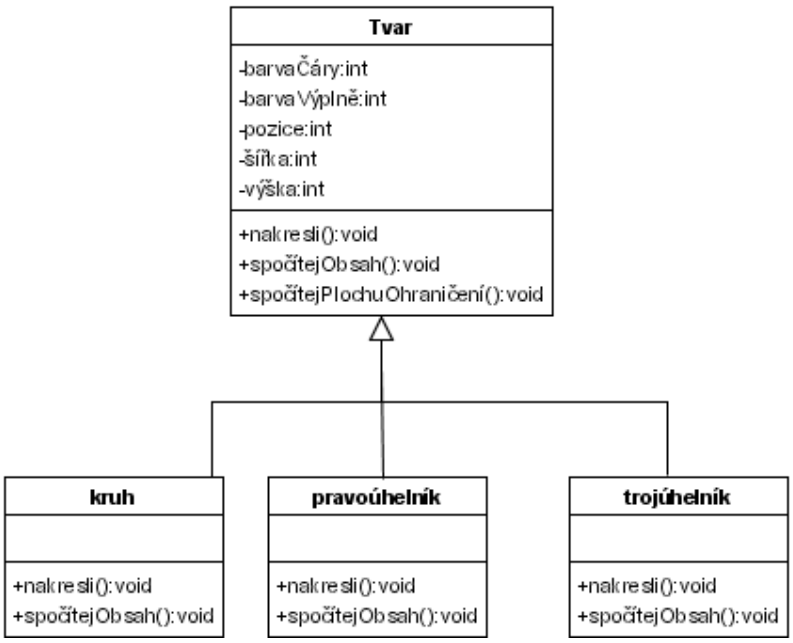
Zobecnění nám umožňuje další stupeň re-use, resp. znovu-použitelnost nikoliv ve vztahu třída a několik jejích instancí, ale třída a od ní několik odvozených tříd.

Pokud namodelujeme několik tříd, které jsou si velmi podobné, mohlo by se jednat o situaci, kde je vhodné použít zobecnění.

ŘEŠENÁ ÚLOHA

Vysvětlení pojmu zobecnění na příkladu

Klasický příklad pro znázornění zobecnění: obecnější třída (předek, nadtřída, bázová třída, předchůdce) se jmenuje tvar a zastupuje nějaký, blíže neurčený dvourozměrný tvar. Tvar má svou barvu čáry, barvu výplně, pozici (pozice těžiště), šířku a výšku, dále má metody nakresli (nakreslí tvar), spočítejObsah (spočítá plošný obsah) a spočítejPlochuOhraničení (spočítá plochu pravoúhelníku ohraničujícího tvar), viz obrázek 3.



Obrázek 3: Třída a podtřídy, zdroj http://mpavus.wz.cz/oo/

Vysvětlení pojmů dědění a specializace na příkladu

Podtřídy (potomci) jsou specializacemi nadtřídy -zde máme kruh, pravoúhelník, trojúhelník. Všude (tj. v jiných diagramech, v kódu,....), kde použijeme instanci (objekt) třídy tvar můžeme použít instanci libovolného jeho potomka. Říkáme, že "potomek je druhem předka" tj. kruh je druhem tvaru, trojúhelník je druhem tvaru - pokud si tuto větu říct nemůžeme, pravděpodobně je ve stromu zobecnění nějaká chyba (opakovaná chyba v učebnicích: předek je bod, potomek je úsečka, přímka, čtverec, kruh,... - zde si nemůžeme říct čtverec je druhem bodu).

Vztahy generalizace-specializace se v konkrétním prostředí realizují pomocí techniky dědění (inheritance). Potomci dědí:

- atributy,
- metody,
- relace,
- omezení.

Potomci mohou ke zděděnému přidávat to svoje (tj. atributy, metody, relace a omezení), dokonce mohou zděděné metody předefinovat (viz níže výklad pro abstraktní metody). To je kruh, pravoúhelník a čtyřúhelník zdědí od předka tvar všechny atributy (barvu čáry, šířku a výšku, atd.), dále zdědí metodu a její kód, např. „Spočítej plochu ohraničení“ a předefinují metody „Nakresli“ a spočítej obsah.

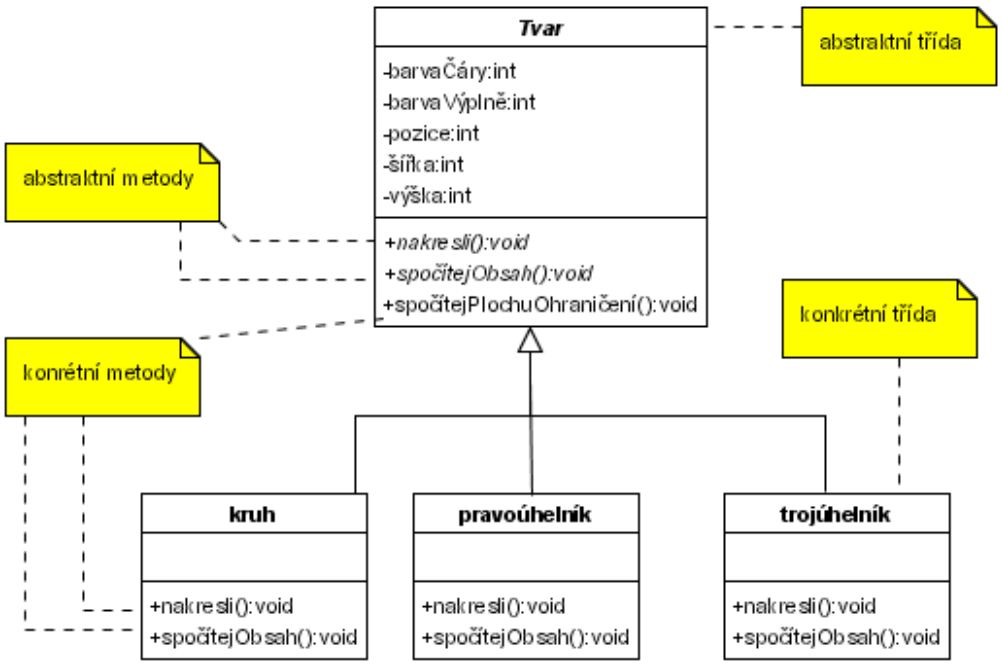
1.2.4 ABSTRAKTNÍ METODY A TŘÍDY

DEFINICE

Metodě, jejíž implementace v předkovi úplně chybí, říkáme abstraktní metoda. Třídě, která má alespoň jednu abstraktní metodu, říkáme abstraktní třída.

Vysvětlení pojmu abstraktní třída na příkladu

V předchozím příkladu jsme si ukázali, že metody předka „spočítejObsah()“ a „nakresli()“ musí být v potomcích předefinovány. Pokud všichni potomci překrývají metodu po svém, tak ani nemá smysl v předkovi tuto metodu implementovat - chceme ji ale přesto v předkovi uvést (můžou nás k tomu vést například důvody uvedené na konci předchozího odstavce). Takovýmto metodám bez implementace říkáme abstraktní metody. Ta třída, která má alespoň jednu abstraktní metodu, není nadefinována do té míry, abychom mohli vytvořit instanci této třídy (abstraktní metoda této třídy je prázdná). Takovéto třídě říkáme abstraktní třída. Metodě, která není abstraktní, pak říkáme konkrétní metoda, třídě, která není abstraktní, pak říkáme konkrétní třída. Předchozí obrázek tedy můžeme zakreslit s využitím notace UML, kde se abstraktní metody a třídy označují kurzívou jako obrázek 4.



Obrázek 4: Abstraktní třídy a metody, zdroj http://mpavus.wz.cz/oo/

1.2.5 POLYMORFISMUS

Pojem polymorfizmu se někdy zbytečně zdá být komplikovaný a těžko pochopitelný, ale jestliže uvažujeme objektově a máme určitou praxi s objektovým modelováním, není tento pojem složitý k pochopení. Polymorfismus znamená mnohotvárnost.

DEFINICE

Podle [Arl2007] jsou polymorfní operace takové operace, které mají mnoho implementací.

Polymorfní operace a metody

Polymorfní metody jsou takové metody, které se vyskytují u více objektů (tj. mají u různých objektů stejnou signaturu), ale mají odlišnou implementaci, dle mpavus.wz.cz/oo/oo-trida-4.php.

ŘEŠENÁ ÚLOHA

Vysvětlení pojmu polymorfní metody na příkladu

Příklad: Máme metody „spočítejObsah()“ a „nakresli()“. V konkrétních třídách mají tyto metody stejnou signaturu, avšak v každé třídě je jiná implementace: metoda „nakresli()“ spuštěná v objektu třídy kruh bude kreslit kruh, metoda „nakresli()“ spuštěná v objektu třídy pravoúhelník bude kreslit čtverec nebo obdélník. Tyto metody jsou tedy polymorfní - v různých třídách mají různé chování.

1.2.6 ZAPOUZDŘENÍ

Softwarové objekty sdílí stejný koncept jako objekty z reálného světa: taktéž obsahují stavy a příbuzné akce. Objekt ukládá ve vlastnostech své stavy a přes metody nabízí příslušné akce. Metody operují nad interními stavy objektu a jsou určeny k prvotnímu způsobu komunikace mezi dvěma objekty.

DEFINICE

Základní kámen OOP je zapouzdření

Skrývání interních stavů a jejich nabízení prostřednictvím metod se nazývá zapouzdření dat – jeden ze základních kamenů objektově orientovaného programování. Viz http://programujte.com/clanek/2007043001-java-tutorial-objektove-orientovane-programovani-3-dil/

SHRNUTÍ KAPITOLY

V této kapitole je čtenář seznámen se základními pojmy OOM, je to stručný úvod do objektově orientovaného modelování. Výklad, vymezení a pochopení těchto pojmů je nezbytnou podmínkou pro porozumění dalších kapitol tohoto učebního textu.

OTÁZKY

Co je to třída?

Vyberte jednu z nabízených možností:

a) Totéž, co objekt.

b) To, co vzniká z objektu při běhu aplikace.

c) Šablona - předpis pro vytvoření objektů**.**

Jaká je správná definice objektu?

Vyberte jednu z nabízených možností:

a) Objekt zajišťuje jen funkcionalitu prvku v informačním systému

b) Objekt je seskupení dat a funkcionality

c) Objekt reprezentuje data pro subjekt v databázi

Při dědění tříd se dědí?

Vyberte jednu z nabízených možností:

a) Atributy i metody.

b) Jen atributy.

c) Jen metody.

Abstraktní metoda u třídy je metoda, jejíž implementace je?

Vyberte jednu z nabízených možností:

a) V předkovi úplně obsažena.

b) V předkovi úplně chybí.

c) V předkovi je obsažena částečně.

Polymorfní metoda se vyskytuje?

Vyberte jednu z nabízených možností:

a) U více objektů, ale má stejnou implementaci.

b) U nadřízeného objektu, ale má odlišnou implementaci u podřízeného objektu.

c) U více objektů, ale má odlišnou implementaci.

ODPOVĚDI

c. b. a. b. c.

---

2 ZÁKLADNÍ ELEMENTY JAZYKA UML

RYCHLÝ NÁHLED KAPITOLY

V této kapitole si objasníme principy jazyka UML. Modelovací jazyk UML je souhrnem grafických notací k vyjádření analytických a návrhových modelů. UML je jazyk, který dává předpoklady a možnosti pro modelování jednoduchých i složitých aplikací s využitím stejné formální syntaxe.

CÍLE KAPITOLY

Cílem kapitoly je objasnit základní pojmy jazyka UML a k čemu tyto diagramy slouží.

ČAS POTŘEBNÝ KE STUDIU

60 minut.

KLÍČOVÁ SLOVA KAPITOLY

UML, Diagramy, metodika RUP, modelování.

2.1 Princip jazyka UML

Charakteristika jazyka UML

Modelovací jazyk UML je souborem grafických notací k vytváření analytických a návrhových modelů informačních systémů. UML dobře poslouží při vytváření jak jednoduchých, tak i složitých aplikací, přitom využívá stejnou formální syntaxi. Je ideálním prostředkem pro sdílení pracovních postupů pro vývojáře, testery, ale i zadavatele a uživatele systému. Slouží rovněž k objasňování požadavků uživatelů na vytvářený systém. Diagramy UML se soustředí vždy na právě jeden pohled na vyvíjený systém. Navrhovaný systém zachycuje celek komplexem jednotlivých diagramů UML.

UML je jazyk pro vizualizaci, specifikaci, stavbu a dokumentaci software zajišťující chod informačních systémů. Používané modelovací techniky s využitím jazyka UML podporují nejznámějšími metodiku UP (Unified Process) a RUP (Rational Unified Process) firmy Rational (nyní ve vlastnictví IBM) a Select Business Solution, viz např. [Kan004].

DEFINICE

UML = Unified Modeling Language (tj. unifikovaný modelovací jazyk)

Unifikovaný: unifikuje Booch, OMT a Objectory modelovací jazyky

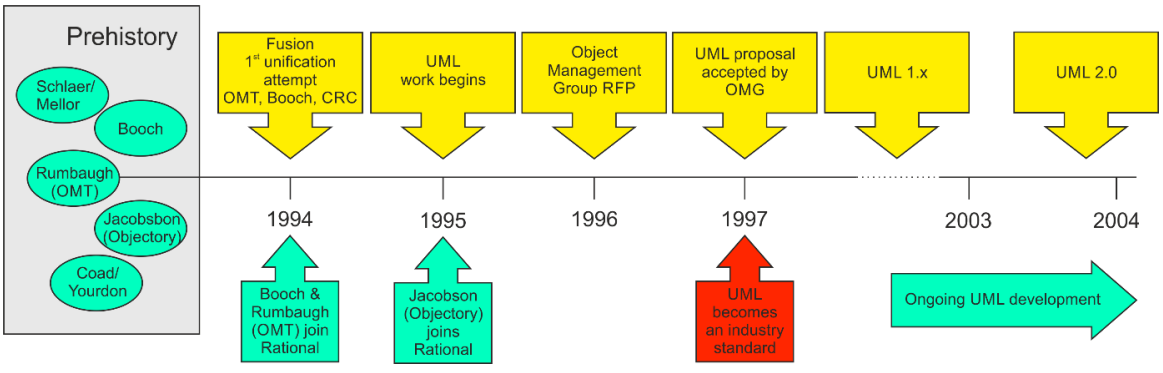
Modelovací: UML je jazyk pro specifikaci, vizualizaci, konstrukci a dokumentaci artefaktů SW systémů.

UML je využitelný i pro business modelování i pro modelování nesoftwarových systémů. V UML lze modelovat jakýkoliv typ aplikace běžící na jakémkoliv typu a kombinaci HW, OS, programovacím jazyku a sítě. Lze modelovat distribuované aplikace.

UML není programovací jazyk, ale je to jazyk, neboť má jasně definovanou syntaxi a sémantiku.

2.2 Historie vzniku jazyka UML

Historie vzniku UML je přehledně zachycena na obrázku 5.



Obrázek 5: Historie vzniku UML, zdroj: [Arl2008)]

Přehled milníků vzniku a tvůrců jazyka UML je uveden a podrobně popsán v [FRA2014].

2.3 UML – skladba

Skladba definuje, z jakých *základních prvků* se UML skládá.

Tyto základní prvky jsou tři:

1) **Předměty (things)**, tj. elementy modelu definující:

- **strukturní abstrakce (structural things)** : podstatná jména (modelu UML), tj. elementy: třída (class), rozhraní (interface), spolupráce (collaboration), případ užití (use-case), aktivní třída (active class), komponenta (component), uzel (node),
- **chování (behavioral things)** : slovesa (modelu UML), tj.: interakce (vztahují se ke komunikaci mezi objekty - např. zprávy (messages), spoje (links)) a stavový stroj (specifikuje sekvenci stavů objektu pomocí stavů, přechodů, událostí a aktivit),
- **seskupení (grouping things)** : balíčky (packages) seskupující (dle potřeby) prvky modelu,
- **poznámky (anotational things)** : poznámky s přidanými informacemi.

2) **Relace (relationships)**, tj. elementy modelu spojující spolu dva (či více) předmětů - **strukturních abstrakcí** (také mohou spojovat **seskupení**); jejich typy jsou:

- **závislost (dependency)** : znázornění vztahu, kdy změnou v jednom elementu je ovlivněn jiný (závislý) element.
- **asociace (association)** : spojení definující vztah mezi elementy,
- **zobecnění (generalization)** : vztahy generalizace-specializace, tj. jeden element je specializací jiného elementu,
- **realizace (realization)** : jeden element je realizací jiného elementu.

3) **Diagramy**: model je souhrn všech předmětů a relací, jejichž znázornění v jednom obrázku by bylo nepřehledné, mnohdy zcela nemožné. Naproti tomu diagram je jeden *pohled*, *okénko*, kterým se díváme na model.

   Rozdělení základních diagramů do dvou skupin:

   Statický model (zaměřený na systémovou strukturu) :

   - diagram tříd
   - diagram komponent
   - diagram nasazení

   Dynamický model (zaměřený na chování systému) :

- diagram případu použití
- sekvenční diagram
- diagram spolupráce
- stavový diagram
- diagram aktivit
- objektový diagram

Zpracováno podle [Arl2007] a http://mpavus.wz.cz/uml/uml-skladba-2.php

2.4 UML – mechanismy

UML má čtyři mechanismy, které se prolínají celým jazykem. Tyto mechanismy jsou čtyři.

1) Specifikace: každý element může (či měl by) být specifikován textem, který popisuje sémantiku tohoto elementu. Tato specifikace upřesňuje, blíže popisuje, udává smysl modelovaného elementu. Popisuje business pravidla elementů (tudíž má největší význam u elementů popisujících problémovou doménu).

2) Ozdoby (adornments): další informace známé o elementu modelu. Každý element může být zadán jednoduchým tvarem, ale je možno přidávat k němu i další informace - ozdoby. Proč je těchto ozdob u elementu zobrazeno někdy více a někdy méně: postupně vytváříme model: zpočátku máme málo informací, které postupně doplňujeme

3) Podskupiny (common division) : udávají, jak je možno rozdělovat (seskupovat) jednotlivé elementy; první způsob dělení:

- **klasifikátor a instance**: pro dva elementy UML objekt a třída platí, že objekt je **instance**, kdežto třída je **klasifikátor**. Podobný vztah klasifikátor-instance lze nalézt pro další elementy UML. Každý element je buď klasifikátor anebo instance. Toto rozlišení je velmi důležité. Osvojením tohoto dělení si usnadníte komunikace mezi členy týmu. Můžeme se s tímto setkat i v CASE nástrojích a v literatuře,
- **rozhraní a implementace**: v pasáži o objektu objekt jsme se zmínili o **protokolu zpráv**, což je rozhraní objektu. Implementace pak jsou metody, které *řeší*, implementují, toto rozhraní.

4) Mechanismy rozšiřitelnosti: jazyk UML sám v sobě obsahuje připravené mechanismy umožňující rozšířit jazyk tak, aby vyhovoval momentálním potřebám. Máme k dispozici tři mechanismy rozšiřitelnosti:

- **omezení (constraints)**
- **stereotypy (stereotypes)**
- **označené hodnoty (tagged values)**

Zpracováno podle [Arl2008].

SAMOSTATNÝ ÚKOL

Důkladně promyslete a zapamatujte si výše uvedené základní prvky a mechanizmy jazyka UML. Prostudujte podrobnější popisy těchto pojmů v [Arl2007] a v [Fra2014]. Konfrontujte s popisy na Internetu.

SHRNUTÍ KAPITOLY

Tato kapitola se věnovala základním pojmům jazyka UML. Naučili jste se k čemu UML slouží a z jakých předpokladů vychází. Tyto základní pojmy je nutno se naučit a pochopit jejich význam. Tyto pojmy se pochopí lépe při druhém čtení po prostudování dalších kapitol a zejména příkladů využití jazyka UML. Proto autor doporučuje se k této kapitole znovu vrátit.

OTÁZKY

Základní rozděleni UML diagramů je:

Vyberte jednu z nabízených možností:

a. Komponenty a správa modulů

b. Statická struktura a dynamické chování

c. Aktivity a nasazení

Co znamená zkratka UML?

Vyberte jednu z nabízených možností:

a) Universal Modeling Language.

b) Unified Modeling Language.

c) Unified Modal Language.

Co je to UML?

Vyberte jednu z nabízených možností:

a) UML je souhrnem především grafických notací k vyjádření analytických a návrhových modelů software.

b) UML umožňuje především programovat SW.

c) UML je jazyk především pro vytváření dokumentace SW.

Co umožňuje UML?

Vyberte jednu z nabízených možností:

a) Popsat objektovou analýzu a návrh SW.

b) Modelovat testování SW.

c) Navrhovat a spravovat uživatelské požadavky v SW.

Je správa požadavků součástí UML?

Vyberte jednu z nabízených možností:

a) Částečně

b) ANO

c) NE

Do diagramů pro modelování struktury patří?

Vyberte jednu z nabízených možností:

a) Sekvenční diagram.

b) Diagram tříd.

c) Diagram případu užití.

Do diagramů pro modelování chování patří?

Vyberte jednu z nabízených možností:

a) Diagram nasazení.

b) Diagram aktivit.

c) Diagram balíčků.

ODPOVĚDI

b. b. a. a. c. b. b.

---

3 POPIS JAZYKA UML – USE CASE

RYCHLÝ NÁHLED KAPITOLY

Tato kapitola se zabývá tzv. případy užití. Případy užití, typové úlohy nebo chcete-li užitné případy. Se všemi těmito překlady originálu „**Use Case“** jsou v odborné literatuře používány. V následujícím textu se přikloníme k terminologii „případ užití“, který odpovídá anglickému originálu.

CÍLE KAPITOLY

Cílem kapitoly je objasnit k čemu případy užití přesně slouží. Případy užití popisují přesně funkcionalitu vytvářeného informačního systému a vymezují tím jednoznačně rozsah prací, které je nutno vykonat od zjištění požadavků, přes analýzu, programování. Testování a uvedení do provozu. Každý případ užití popisuje jeden ze způsobů chování systému, jednu jeho požadovanou funkčnost.

ČAS POTŘEBNÝ KE STUDIU

60 minut

KLÍČOVÁ SLOVA KAPITOLY

Use Case, Aktér, Případ užití, Scénář.

3.1 Úvod do problematiky USE CASE

Případy užití, typové úlohy nebo také užitné případy. Se všemi těmito překlady originálu „Use Case“ se setkáváme v odborné literatuře setkat. V následujícím textu se přikloníme k terminologii „případ užití“, který odpovídá anglickému originálu.

Vývojáři bez ohledu na svou orientaci na vývojové prostředí se již dlouhou dobu snaží modelovat typické interakce uživatelů se systémy, aby co nejlépe pochopili skutečné požadavky uživatelů na budoucí systém a zároveň aby vymezili rozsah navrhované aplikace.

Případy užití popisují funkčnost a vymezují rozsah prací IS

Případy užití zachycují přesně funkčnost, která bude budoucím informačním systémem pokryta a vymezují tak jednoznačně rozsah prací. Každý případ popisuje jeden ze způsobů systému, popisuje tedy jednu jeho požadovanou funkčnost. Pro co nejnázornější výklad pojmu USE CASE a pro výklad UML diagramů v dalších kapitolách si popíšeme konkrétní modelový případ návrhu části informačního systému „Objednávka na e-shopu“. Vytvoříme tzv. případovou studii, kdy budeme navrhovat část elektronického objednávání zboží, zkráceně e-shop“.

3.2 Případová studie USE CASE

Text tohoto výukového materiálu provází případová studie „Objednávka na e-shopu“. V této kapitole si objasníme celou problémovou oblast vytváření „USE CASE“ diagramu a k němu příslušné scénáře.

Cílem bylo demonstrovat praktické použití UML na konkrétním projektu s cílem zjednodušit a zprůhlednit ukázky v této případové studii. Tato studie řeší pouze část funkcionality nakupování po internetu – e-shopu. Neřeší modelování kompletního e-shopu pro složitost a rozsáhlost problematiky, která by způsobila nepřehlednost analýzy. Z tohoto důvodu byla ponechána stranou i důležitá část informačního systému, a to odstraňování a archivace dat.

Ukázky by měly demonstrovat základní metody UML a tomu je přizpůsobena složitost případové studie i uvedených příkladů.

Cílem není dokonalá analýza dané problematiky, ale demonstrace technik UML, a proto na některých místech učebního textu jsou uvedeny i ukázky, které nepochází z modelové studie.

PŘÍPADOVÁ STUDIE

Příklad z praxe případové studie

Případová studie předpokládá modelovou situaci, kdy softwarová firma získala zakázku na analýzu, návrh a vývoj informačního systému internetové aplikace e-shop. Představme si tedy situaci, kdy existuje poptávka po vytvoření aplikace e-shop a softwarová firma má za úkol tuto aplikaci navrhnout a vytvořit. To sebou nese celou řadu problémů týkající se oblasti oborů marketingu a projektového managementu. My se zaměříme pouze na část e-

shopu, a to na návrh objednávkového systému na internetu z pohledu analýzy při návrhu software. Pro jednoduchost budeme v popisované modelové studií uvažovat pouze o jednom modulu informačního systému, a to modulu e-shopu pro objednávání zboží zákazníkem. Záležitosti ekonomické (účetnictví, fakturace) a další funkce e-shopu jako jsou například reklamní kampaně, hodnocení efektivity a další funkce, nejsou předmětem případové studie.

Vedoucí pracovníci softwarové firmy absolvovali několik úvodních jednání a porad s analytiky, kteří mají navrhnout aplikaci e-shop a připravit ji k programování. Výsledky těchto jednání byly sumarizovány do podoby uživatelských požadavků a představy o fungování budoucí aplikace.

Pro pochopení souvislostí uvedeme nyní seznam požadavků na funkce e-shopu:

1) Zadávání a evidence zboží s funkcemi

Základními údaji jsou název, kód a cena a popis zboží pomocí HTML editoru. Definice cen poskytuje možnost vytváření až 30 cenových skupin, možnosti vytváření variant zboží, zboží je možné vložit do různých kategorií, samozřejmě lze vkládat neomezený počet fotografií, či jiných příloh, vestavěný export a import pomocí CSV, XML).

2) Definice způsobů dopravy a platby

V e-shopu budou pokryty všechny obvyklé způsoby platby – platba bankovní kartou, hotovostní platba při odběru zboží, platba převodním příkazem a splátkový prodej.

3) Zákazníci, zvýhodněné ceny, cenové skupiny

Uživatelé nakupující v e-shopu mají možnost volby, zda se zaregistrují nebo ne. Lze nakoupit i bez registrace. Každému zaregistrovanému uživateli lze nastavit slevu v procentech i cenovou skupinu, podle obratu zákazníka.

4) Statistiky

Interní statistiky návštěvnosti, historie pageranku, + napojení na Google Analytics, Google pro Webmastery, Toplist, Navrcholu, statistiky objednávek, obratů, statistiky prodeje a návštěvnosti pro jednotlivé produkty, napojení pro měření konverzí pro libovolné systémy.

5) Vytvoření rozhraní pro napojení ekonomického systému

V pěti bodech je shrnuta celá funkcionalita e-shopu. Pro naše účely vysvětlení pojmů a diagramů modelovacího jazyka UML se dále budeme zabývat jen objednáváním zboží na e-shopu z pohledu zákazníka.

ROLE V PŘÍPADOVÉ STUDII E-SHOP OBJEDNÁVKY ZÁKAZNÍKA

1) Zákazník (registrovaný a neregistrovaný)

Tři role v případové studii

Neregistrovaný zákazník má volný vstup na stránky e-shopu a není omezen z pohledu tvorby objednávky, ale pouze její modifikací. Tento typ uživatele může zboží vyhledávat, prohlížet, popřípadě přidávat do košíku, vyplnit a následně odeslat objednávku na zboží. Nemá nárok na většinu slev.

Registrovaný zákazník má možnost spravovat svůj účet, kde si může prohlížet svou historii objednávek, modifikovat objednávku před její expedicí a hlavně má možnost uplatňovat slevy dle obratu.

Systém pro elektronickou platbu: příjme zákazníkův požadavek platit a prostřednictvím vybrané služby (kreditní karta, paypal) mu to umožní.

Systém pro registraci: slouží pro registraci, Přihlášeni a ověření totožnosti zákazníka.

2) E-shop (systém)

Provádí případnou registraci návštěvníka. Zajišťuje přístup ke katalogům zboží. Slouží pro zobrazování požadavků zákazníka, provází jej nákupem a odkáže ho v případě zájmu na platbu. Bude vytvořeno uživatelské prostředí, které napomáhá k jednoduššímu nákupu zboží.

3) Platba (systém)

příjme zákazníkův požadavek platit a prostřednictvím vybrané služby (kreditní karta, hotovost, paypal).

DIAGRAM PŘÍPADU UŽITÍ V PŘÍPADOVÉ STUDII

V diagramu případu užití jsou zobrazeny 3 aktéři: Neregistrovaný, Registrovaný a E-shop (systém).

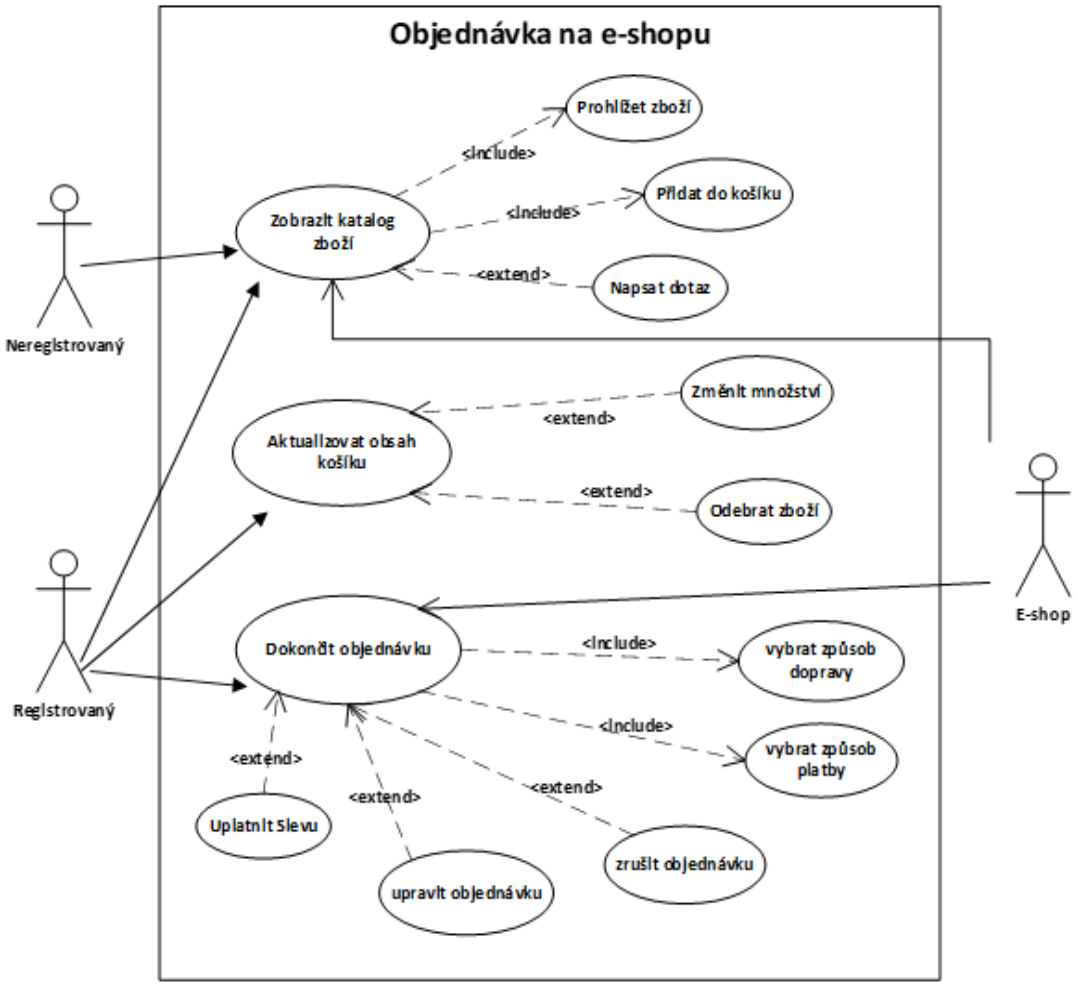
Neregistrovaný:

Nemá povinnost se registrovat do systému, ale může objednávat zboží. Jeho nevýhodou je nemožnost uplatnění slevového programu a při každém vstupu na portál e-shopu vyplnit kontaktní údaje.

Registrovaný:

Objednává zboží, na které může uplatnit slevový program dle obratu, může sledovat svou historii nákupů.

E-shop (systém): Zajišťuje vstup zákazníků na portál e-shopu, registraci zákazníka a přihlášení zákazníka. Provází zákazníka při tvorbě objednávky, zajišťuje požadavek na způsob placení.



Obrázek 6: Objednávka na e-shopu, příklad případu užití, zdroj vlastní

3.3 Scénáře v případech užití

Scénář se skládá z postupných kroků, které zachycují vzájemné akce mezi uživatelem (aktérem) a obrazovkou (systémem)

Případy užití jsou základním elementem při plánování, vývoji a realizaci projektu – vytváření informačního systému. Co znamená pojem **scénář případu užití**? Scénář se skládá z postupných kroků, které zachycují vzájemné akce mezi uživatelem (aktérem) a obrazovkou (systémem). V systému z případové studie „Objednávky v e-shopu“, můžeme vytvořit scénář, viz tabulka 1:

Krok	Role	Akce
1	Zákazník	Zadání www adresy e-shopu vstoupí na úvodní stránku e-shopu.
2	Zákazník	Zadáním přihlašovacích údajů se přihlásí se do systému.
3	Systém	Zkontroluje údaje a zobrazí přehledové informace o zákazníkovi a nabídne katalog.
4	Zákazník	Pomocí rolování nebo funkce „Vyhledat“ vyhledává vhodné zboží.
5	Systém	Ke zvolenému zboží zobrazí základní informace, alternativně podrobné informace, údaje o možné slevě, zobrazí obrázky ke zboží.
6	Zákazník	Studuje popisy a údaje ke zboží.
7	Zákazník	Ke zvolenému zboží zadá množství.
8	Systém	Zvolené zboží vloží do košíku a dotáže se se, zda chce dále pokračovat ve výběru nebo přejít k zaplacení.
9	Zákazník	Nakupující buď pokračuje ve výběru zboží (body 4 – 8) nebo se rozhodne zaplatit.
10	Systém	Zkontroluje košík, provede výpočty, vypočítá slevu dle obratu, sestaví návrh objednávky a požádá zákazníka o odsouhlasení objednávky
12	Zákazník	Potvrdí objednávku, případně provede úpravy v nákupním košíku a znovu požádá systém o kompletaci objednávky.
13	Systém	Nabídne zákazníkovi na obrazovce způsob platby a dopravy.
14	Zákazník	Zvolí si z příslušných možností metodu platby kartou a dopravu PPL.
15	Systém	Zaeviduje objednávku a požádá zákazníka o kontrolu a potvrzení
16	Zákazník	Potvrdí nebo opraví dodací údaje a potvrdí objednávku.
17	Systém	Objednávku zaeviduje a zašle e-mail s objednávkou a dalšími informacemi.
18	Systém	Zboží je zasláno zákazníkovi.

Tabulka 1: Scénáře případu užití registrovaný uživatel, který uplatňuje slevu a platí kreditní kartou, zdroj: vlastní

Krok	Role	Akce
1	Zákazník	Zadání www adresy e-shopu vstoupí na úvodní stránku e-shopu.
2	Zákazník	Hledá v katalogu zboží na e-shopu.
3	Systém	Ke zvolenému zboží zobrazí základní informace, alternativně podrobné informace, a zobrazí obrázky ke zboží.
4	Zákazník	Prohlíží údaje o zboží.
5	Zákazník	Zvolí si zboží a zadá požadované množství.

6	Systém	Vybranou položku s množstvím vloží do košíku a dotáže se, zda chce pokračovat ve výběru dalšího zboží nebo přejít k zaplacení.
7	Zákazník	Zákazník buď pokračuje ve výběru zboží (body 4 – 8) nebo se rozhodne zaplatit.
8	Systém	Zkontroluje košík, provede výpočty, sestaví návrh objednávky a požádá zákazníka o odsouhlasení objednávky.
9	Zákazník	Potvrdí objednávku, případně provede úpravy v nákupním košíku a znovu požádá systém o kompletaci objednávky.
10	Systém	Nabídne zákazníkovi na obrazovce způsob dopravy a platby.
11	Zákazník	Zvolí si metodu platby (například z účtu) a jak zboží dopravit.
12	Systém	Zaeviduje objednávku a požádá zákazníka o potvrzení a pokud došlo ke změně dodacích údajů o jejich opravu.
13	Zákazník	Potvrdí nebo opraví dodací údaje a potvrdí objednávku.
14	Systém	Zavede objednávku do databáze a potvrdí na e-mail zákazníka včetně údajů k platbě z účtu, tj. účet dodavatele a variabilní symbol.
15	Zákazník	Zaplatí za objednávku převodem na účet.
16	Systém	Počká na potvrzení došlé platby z ekonomického systému a pak odešle zboží zákazníkovi a informuje zákazníka mailem.

Tabulka 2: Scénář případu užití „neregistrovaný uživatel“, který platí přes bankovní účet, zdroj: vlastní

Stojí za povšimnutí, že takto zapsaný scénář případu užití je podobný se skutečnými scénáři filmů nebo divadelních her.

Dostáváme se k odpovědi na otázku „Co je to definici případu užití“. **Případ užití je soubor scénářů, které dohromady spojuje společný cíl.**

3.4 Popis případu užití

Jak má vypadat popis případu užití? Modelovací jazyk UML pro psaní scénářů nedefinuje žádný standard, existují však doporučení osvědčených v praxi:

- Název případu užití by měl být tvořen pomocí slovesné vazby (představuje nějakou akci v systému, tedy např. *nákup na e-shopu z pohledu zákazníka*.
- Pro zápis hlavního úspěšného scénáře použijte jednoduchou sekvenci číslovaných kroků, viz například tabulka 3.
- Pro zápis případných rozšíření v podobě alternativních scénářů použijte opět číslovanou sekvenci odkazující se na hlavní sekvenci.
- Používáme stručné a srozumitelné věty.

Použití těchto konstruktů ilustruje tabulka 3.

Krok	Role	Akce
1	Zákazník	ZAČÁTEK CYKLU Vybere zboží z nabízeného seznamu

2	Systém	Zobrazí formulář nabízeného zboží s podrobnou specifikací
3	Zákazník	POKUD Zákazník souhlasí s nabízenými parametry zboží, vloží zboží do nákupního košíku JINAK Přejde zpět na seznam zboží KONEC - POKUD
	Systém	Zobrazí seznam nabízeného zboží
5	Zákazník	Pokračuje ve výběru zboží KONEC CYKLU

Tabulka 3: Případ užití: Výběr zboží v e-shopu s uplatněním podmínky, zdroj: vlastní

3.5 Vstupní a výstupní podmínky

Pre-Conditions a Post-Conditions

Kromě uvedeného popisu případu užití formou scénářů je možné popis dále upřesnit pomocí přídavných sekcí. Takovými sekcemi mohou být **vstupní podmínky případu užití** (Pre-Conditions), definující předpoklady, které musí být splněny, aby případ užití mohl být zahájen, nebo naopak **výstupní podmínky** (Post-Conditions), určující kritéria, která musí být splněna po skončení případu užití.

Podrobnější popis vstupních a výstupních podmínek byl zpracován v předchozím elektronickém skriptu [Fra2014], které je k dispozici studentům předmětu „Objektové metody modelování“ na e-learning portále SU v rámci připraveného kurzu.

3.6 Postup tvorby popisu případu užití

K postupu tvorby popisu případu existuje v literatuře celá řada doporučení, doplnění a upřesnění.

SAMOSTATNÝ ÚKOL

Prostudujte si tato doporučení v literatuře, zejména v [Arl2007] a [Fra2014]. Konfrontujte nabyté poznatky se zdroji na Internetu.

SHRNUTÍ KAPITOLY

Kapitola se zabývá 1. diagramem UML, a to USE CASE diagramem. Popisuje jeho prvky a vazby mezi nimi. Na příkladu vysvětluje jeho možnosti využití. Kapitola dává návod, jakým způsobem postupovat při vytváření USE CASE a jak se vytváří vstupní a výstupní podmínky.

OTÁZKY

Který výrok o případech užití USE CASE není pravdivý?

a) Případy užití zachycují přesně funkčnost, která bude informačním systémem pokryta a vymezují tak jednoznačně rozsah prací.

b) Případ užití popisuje více způsobů užití funkčnosti systému.

c) USE CASE je součástí UML.

Které dva prvky jsou součástí USE CASE diagramu?

a) Actor a use case.

b) Actor a činnost.

c) Činnost a rozšíření.

Které dvě spojnice se používají v USE CASE diagramu?

Vyberte jednu z nabízených možností:

a) Extend a Include.

b) Extend a Output.

c) Output a Input.

Diagramy případu užití se používají k modelování?

Vyberte jednu z nabízených možností:

a) Posloupnosti zpráv předávaných mezi objekty.

b) Základních struktur v systému.

c) Interakce uživatele se systémem.

Dvěmi základními prvky „USE CASE“ diagramu jsou?

Vyberte jednu z nabízených možností:

a) Aktér (znázorňovaný jako postavička s názvem) a případ užití (znázorňovaný jako elipsa s názvem uvnitř)

b) Aktér (znázorňovaný jako kruh s názvem uvnitř) a případ užití (znázorňovaný jako elipsa s názvem uvnitř)

c) Aktér (znázorňovaný jako postavička s názvem pod) a případ užití (znázorňovaný jako kruh s názvem uvnitř)

ODPOVĚDI

b. a. a. c. a.

---

4 POPIS JAZYKA UML – MODELOVÁNÍ TŘÍD A OBJEKTŮ

RYCHLÝ NÁHLED KAPITOLY

Tato kapitola se zabývá a popisuje základní diagram statické struktury IS, tj. modelování tříd a objektů. Dále se zabývá vztahy mezi objekty. Toto téma řeší diagram UML tříd a diagram UML objektové spolupráce.

CÍLE KAPITOLY

Cílem kapitoly je vysvětlit pojmy třída a objekt a naučit se porozumět a vytvořit model objektové spolupráce.

ČAS POTŘEBNÝ KE STUDIU

90 minut

KLÍČOVÁ SLOVA KAPITOLY

Model, IS, třída, objekt, objektová spolupráce.

4.1 Základní popis diagramu tříd

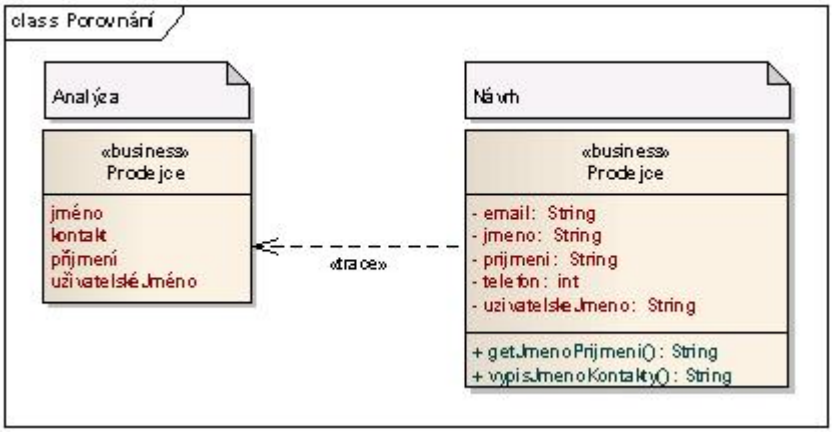
DEFINICE

Diagram tříd (Class Diagram) představuje „statický pohled na modelovaný systém“ [Buch2007] a jeho úkolem je znázornit typy objektů v systému a jejich vztahy. [Fow2003] Návrh tříd, jejich odpovědností a následné vytvoření tohoto diagramu je jedním z prvních a základních kroků analýzy navrhovaného programového systému. [Buch2007]

Při tvorbě diagramu tříd je nutné vzít v úvahu jeho účel a rozlišit, zda potřebujeme vyjádřit požadavky na modelovaný software nebo získat podrobný popis designu, atd. Z tohoto důvodu se dle [Buch2007] rozeznávají tři úrovně modelu tříd – konceptuální, designová a implementační.

Konceptuální model je vytvářen za účelem analýzy požadavků na software

Konceptuální (doménový, analytický) model (viz. Obrázek 7) je vytvářen za účelem analýzy požadavků na software. Obsahuje pouze tzv. byznys třídy (business classes), které modelují problémovou oblast a jsou tedy součástí slovníku problémové domény (slovníčku pojmů). [Buch2007] U jednotlivých tříd se uvádí obvykle jen názvy klíčových atributů a některých klíčových metod. Pokud je diagram vytvářen pouze za účelem znázornění relací mezi třídami, je možné atributy i metody vynechat. [Arl2008].



Obrázek 7: Porovnání třídy analytického a návrhového modelu tříd, zdroj: http://uml.czweb.org/diagram_trid.htm

Model návrhu

Designový model (model návrhu) vychází z modelu konceptuálního, který rozšiřuje a zpřesňuje například o viditelnosti atributů a metod, datové typy apod. Dále do modelu přidává třídy uživatelského rozhraní (presentation classes) a třídy obsluhující systémové události (control classes), [Buch2007]. Z jedné třídy v analytickém modelu se tedy může

stát v designovém modelu více návrhových tříd. Mezi analytickými třídami a třídami návrhu existuje relace typu trace, viz obrázek 7, [Arl2008].

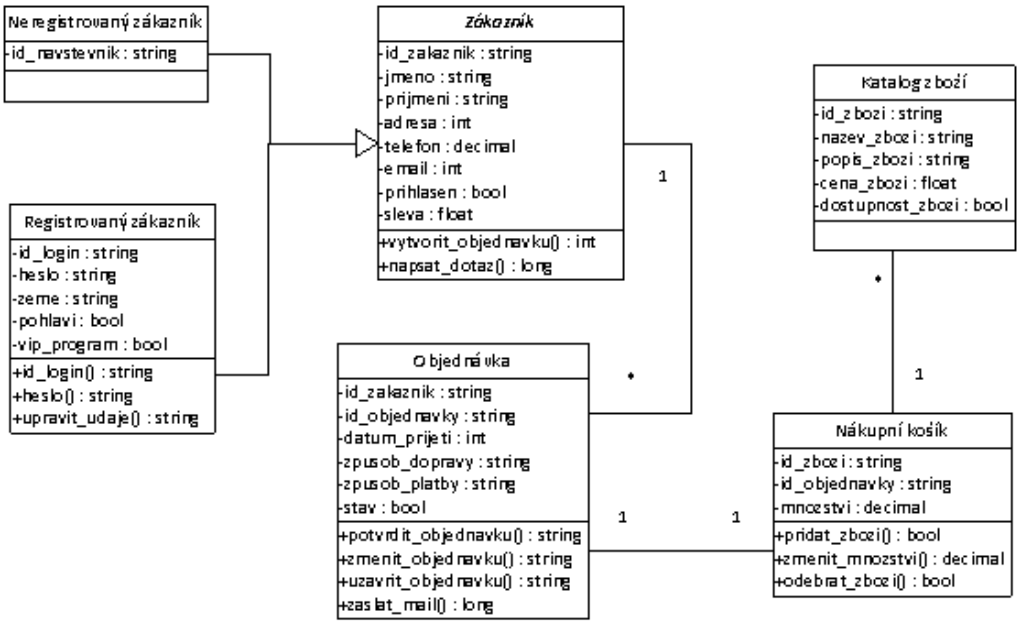
Implementační model se zaměřuje na „grafické zobrazení implementovaného kódu“, [Buch2007].

4.2 Prvky diagramu tříd a doporučení k vytváření modelu tříd

Mezi prvky používané v diagramu tříd lze zařadit třídy (classes), asociace (associations), rozhraní (interfaces) a popřípadě balíčky (packages). [Buch2007] Třída je „abstrakcí objektů se stejnými vlastnostmi, stejným chováním a stejnými vztahy k ostatním objektům.“ [Buch2007] Každá třída má popsány vlastnosti a operace (souhrnně označovány jako features), které může provádět, a omezení definující, jak mohou být jednotlivé třídy propojeny. [Fow2003] Vlastnosti tříd jsou v UML označovány jako atributy, operace ve fázi návrhu jako metody (V rámci analýzy se používá označení operace). Třídy jsou vzájemně propojeny pomocí asociací. [Buch2007]; [Arl2008]

Pro vytváření modelu tříd pomocí diagramu UML platí celá řada doporučení. Nejdůležitějším pravidlem je, že odpovědnosti tříd, které se zaznamenávají jako atributy, by měly být identifikovány již v analytickém (doménovém) modelu. Pro pojmenování tříd, metod, atributů platí další sada pravidel, které souvisí s konvencemi použitého programovacího jazyka. Další pravidla platí pro vazby mezi třídami, důraz na jejich minimalizaci, atd. Tato pravidla jsou důležitá pro pochopení problematiky a jsou podrobně rozpracována v literatuře, viz např. [Clas2006b], [Arl2008] a [Fra2014]. Návrh a vykreslení tohoto diagram je velmi obtížný proces, vyžadující zkušenosti a v praxi se postupuje tak, že se diagram diskutuje v týmu a postupně (iteračně) se diagram tříd vylepšuje, než se dosáhne konečné verze.

4.3 Příklad modelu tříd objednávka na e-shopu



Obrázek 8: Doménový model tříd pro objednávku na e-shopu, zdroj: vlastní

4.4 Model objektové spolupráce

Cílem této podkapitoly je naučit se porozumět a vytvořit model objektové spolupráce. Výše jsme ukázali, jak pomocí analytických tříd můžeme modelovat statickou strukturu systému. Pro identifikaci tříd v systému posloužily vytvořené případy užití. Pro vytváření modelu objektové spolupráce využijeme opět případy užití k tomu, abychom pro ně hledali jejich realizace, tedy takové množiny tříd, které provádějí chování případu užití pomocí vzájemné spolupráce. Jinak řečeno se jedná o převod slovního popisu scénáře případu užití na model interakce identifikovaných tříd.

Proces hledání realizace případů užití je soustavným (iteračním) upřesňováním. Přitom procházíme specifikace jednotlivých případů užití a modelujeme způsob, jak požadované chování zajistit pomocí nalezené množiny analytických tříd a jimi poskytovaných operací. Volání těchto operací je zajišťováno systémem předávání zpráv mezi objekty.

Pro modelování spolupráce objektů používáme zvláštní typy diagramů, které mají vyjadřovací schopnosti k tomu, aby znázornily, jak mezi sebou objekty spolupracují. Právě interakční diagramy jsou tím nástrojem, který nám pomůže odhalit většinu operací spolupracujících tříd. [Kan004]

K modelování spolupráce objektů používáme zvláštní typy diagramů

4.5 Základní charakteristika a prvky objektového diagramu

Objektový diagram (Object Diagram) je snímkem objektů a jejich vztahů v systému v určitém časovém okamžiku. [Buch2007] Je také nazýván diagramem instancí z důvodu, že zobrazuje instance tříd. Používá se především pro znázornění určité konfigurace objektů či zobrazení vzájemně propojených objektů ve speciálních situacích, kdy je diagram tříd či sekvenční diagram nepostačující. [Buch2007]; [Fow2003] Objektový diagram může být chápán jako speciální případ diagramu tříd vytvářený za účelem zdůraznit vazby mezi instancemi.

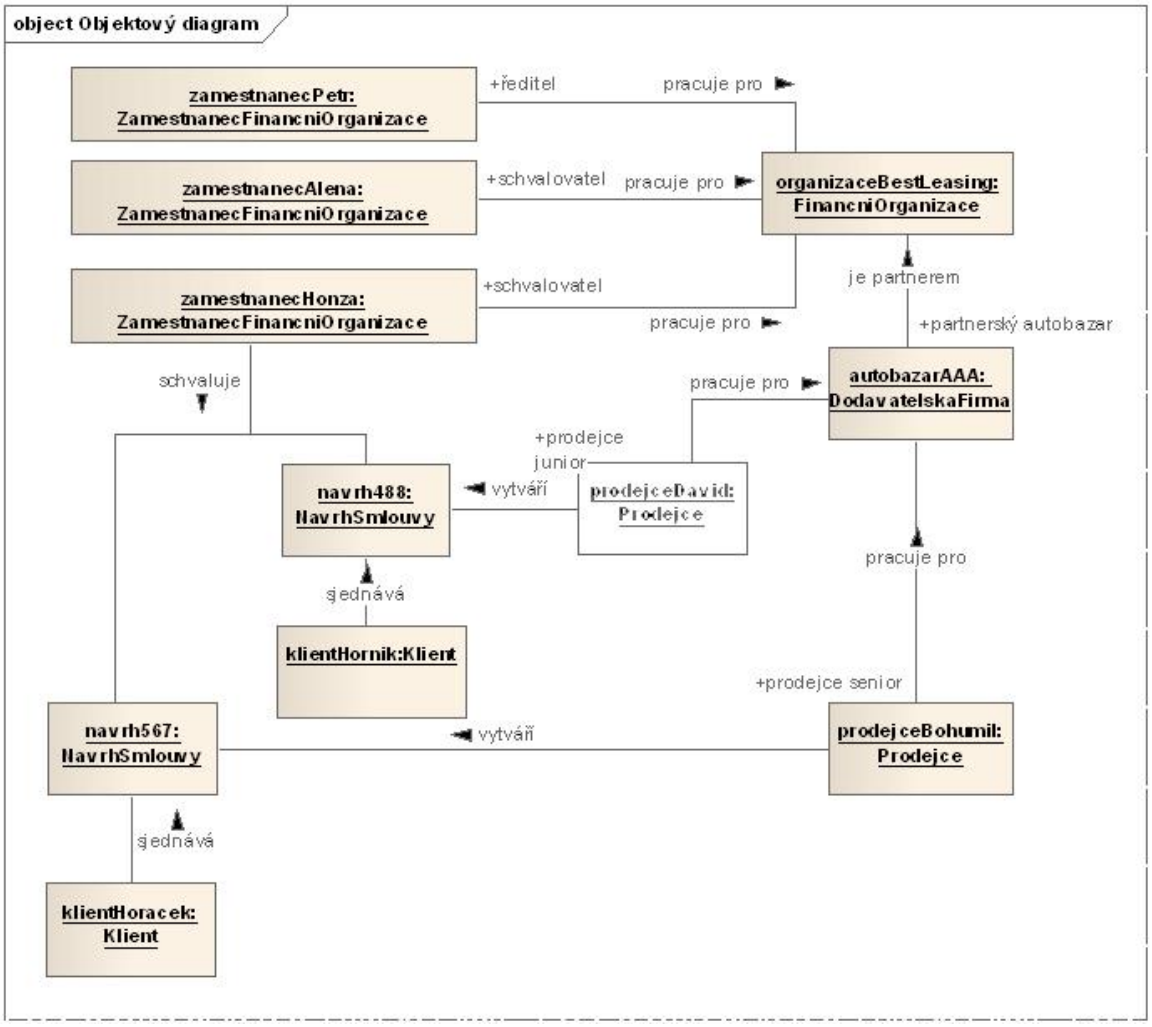
Objektový diagram je užitečný také v počátečních fázích projektu pro modelování ukázek problémové domény, které odhalují objekty a jejich vztahy (viz. Obrázek 9). Často se také používá pro modelování testovacích případů (test cases) pro ověření správnosti diagramu tříd. [Pen2003]

Objektový diagram se svou notací velmi podobá diagramu tříd či komunikace a obvykle obsahuje pouze objekty (objects) a spojení (connections) mezi nimi. Atributy se u objektů vyznačují pouze v případě, že je to nutné pro jejich jednoznačnou identifikaci, metody se neuvádějí vůbec, viz [Obj2006].

SAMOSTATNÝ ÚKOL

Vyzkoušejte si tvorbu objektového diagramu na tématu v rámci své seminární práce, využijte k tomu sadu podrobných návodů a doporučení zpracovaných v publikacích [Fra2013], [Arl2008] a [Pen2003].

4.6 Příklad objektového diagram



Obrázek 9: Objektový diagram, zdroj: http://uml.czweb.org/diagram_trid.htm

SHRNUTÍ KAPITOLY

Tato kapitola vysvětluje pojmy objekty a třídy. Jsou v ní obsaženy doporučení pro jejich vytváření. V této souvislosti byla probrána metodika vytváření objektového diagramu.

OTÁZKY

Na kterých dvou principech je založena struktura tříd?

Vyberte jednu z nabízených možností:

a) Zodpovědnost a zapouzdření.

b) Struktura a metodika.

c) Název atributu a viditelnost.

Který z pojmů nevyjadřuje vztah mezi třídami?

Vyberte jednu z nabízených možností:

a) Agregace.

b) Rekurze.

c) Asociace.

Třída je standardní konstrukce UML používaná pro definování vzorů, z nichž během běhu systému (software) vzniká(jí)?

Vyberte jednu z nabízených možností:

a) Programový kód.

b) Objekty.

c) Události v systému.

Diagram objektové spolupráce klade důraz především na?

Vyberte jednu z nabízených možností:

a) Zjednodušený přehled jednotlivých funkcí systému.

b) Na pořadí jednotlivých událostí.

c) Kontext a uspořádání spolupracujících objektů.

Se kterým diagramem úzce souvisí objektový diagram?

Vyberte jednu z nabízených možností:

a) S diagramem nasazení.

b) Se stavovým diagramem.

c) S diagramem tříd.

ODPOVĚDI

a. b. b. c. c.

5 POPIS JAZYKA UML – STAVOVÉ DIAGRAMY

5.1 Základní charakteristika

Stavový diagram (State Machine Diagram) „zachycuje jednotlivé stavy objektu a přechody mezi nimi.“ [Buch2007] Stavové diagramy se používají především pro popis chování určitého objektu napříč více případy užití a jejich vznik je spojen už s prvními objektově orientovanými technikami. [Fow2003]

Stavové diagramy jsou jednou ze známých technik pro znázornění chování systému. Popisují všechny možné stavy, které může nabývat konkrétní objekt systému, jinými slovy modelují chování objektu napříč všemi případy užití. Zároveň znázorňují, jak se stavy objektu mění v závislosti na událostech, které se ho dotýkají. [Kan2004]

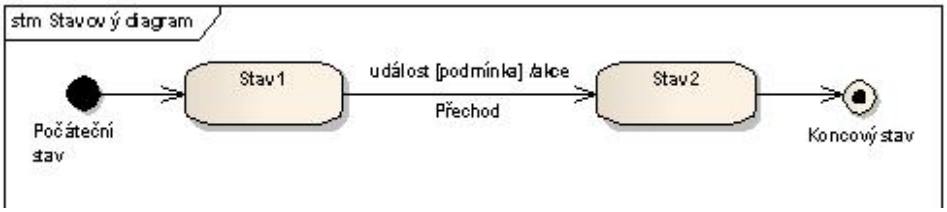
Stavové diagramy jsou jednou ze známých technik pro znázornění chování systému

V mnoha objektově orientovaných technikách se stavové diagramy kreslí pro konkrétní třídu s cílem zachytit životní cyklus konkrétního objektu.

5.2 Prvky stavového diagramu

Základními prvky stavového diagramu jsou stavy, přechody a události, viz obrázek 10. Pokud to CASE nástroj umožňuje, může být diagram ohraničen rámem s názvem objektu. V případě, že se stavy nepohybují v cyklu, měl by diagram obsahovat počáteční (initial state) a koncový stav (final state). [Arl2008]

Stav (state) je dle [Rum2004] „situace v životě objektu, během níž objekt splňuje nějakou podmínku, provádí nějakou operaci nebo čeká na událost.“



Obrázek 10: Prvky stavového diagramu,
zdroj: http://uml.czweb.org/diagram_trid.htm

Přechody (transitions) představují podmínky pro přechod objektu z jednoho stavu do druhého. V diagramu jsou značeny linií vedoucí od jednoho stavu k druhému. Jejich popis se skládá ze tří základních částí: událost [podmínka]/ akce. K vykonání uvedené akce a přechodu do dalšího stavu může dojít pouze v případě, pokud je při vzniku události uvedená podmínka (guard) pravdivá. Událost (trigger signature) je „specifikací určitého výskytu něčeho v čase a prostoru.“ [Arl2008] Pokud není v diagramu uvedena, znamená to,

že přechod do dalšího stavu probíhá automaticky. Na obrázku 11 jsou přechody označeny pouze událostmi (např. posuzování zahájeno). [Arl2008]; [Fow2003]

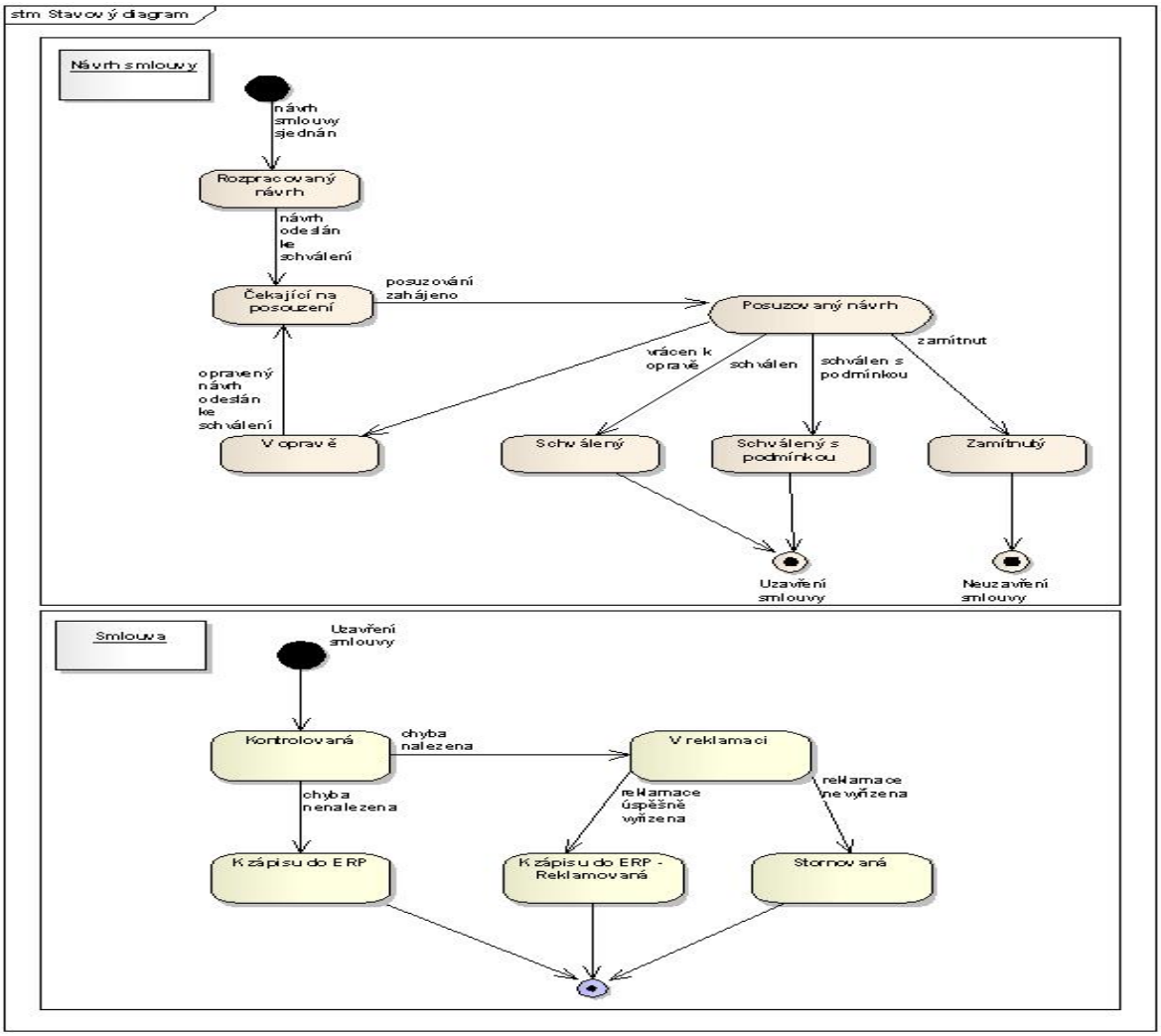
5.3 Doporučení k vytváření stavového diagramu

Podrobnější popis doporučení k vytváření stavového diagramu je zpracován v předchozím elektronickém skriptu [Fra2014], které je k dispozici studentům předmětu „Objektové metody modelování“ na e-learning portále SU v rámci připraveného kurzu.

SAMOSTATNÝ ÚKOL

Podle literatury [Fra2014], [Sta2006] a [Arl2008] vypracujte návod a doporučení pro vytváření stavových diagramů. Získané poznatky aplikujte na svůj příklad stavového diagramu v rámci seminární práce.

5.4 Příkladový stavový diagram



Obrázek 11: Stavový diagram, zdroj: http://uml.czweb.org/

SHRNUTÍ KAPITOLY

V této kapitole byl popsán stavový diagram. Na příkladově je vysvětlen princip a smysl stavového diagramu.

OTÁZKY

K čemu slouží stavový diagram?

Vyberte jednu z nabízených možností:

a) Modeluje chování objektu pro některé případy užití.

b) Popisuje některé stavy, které může nabývat konkrétní objekt systému.

c) Znázorňuje, jak se mění stavy objektů v závislosti na událostech.

Stavový diagram obsahuje prvky?

Vyberte jednu z nabízených možností:

a) Start, stop, stav.

b) Start, stop, příkaz.

c) Start, stav, rozhodnutí.

Kolik může být symbolů „Start“ ve stavovém diagramu?

Vyberte jednu z nabízených možností:

a) Více.

b) Právě jeden.

c) Dva.

ODPOVĚDI

c. a. b.

6 POPIS JAZYKA UML – DIAGRAMY AKTIVIT

RYCHLÝ NÁHLED KAPITOLY

Tato kapitola objasňuje princip diagramu aktivit.

CÍLE KAPITOLY

Cílem je porozumět a naučit se vytvářet diagram aktivit

ČAS POTŘEBNÝ KE STUDIU

90 minut

KLÍČOVÁ SLOVA KAPITOLY

UML, diagram aktivit,

6.1 Základní charakteristika diagramu aktivit

Diagram aktivit (Activity Diagram) je typem diagramu interakcí, který se používá pro popis procedurální logiky, byznys procesů či pracovních postupů. Umožňuje také graficky modelovat jednotlivé případy užití jako posloupnost akcí. [Fow2003] [Arl2008]

Diagramy aktivit představují pravděpodobně nejobtížnější část jazyka UML. Jsou užitečné zejména tím, že dovolují popisovat chování, které má charakter paralelního zpracování.

Diagram aktivit (Activity diagram) zobrazuje sekvenci činností, které podporují jak sekvenční, tak paralelní chování. Diagram aktivit je v podstatě variantou stavového diagramu.

Activity diagram zobrazuje sekvenci činností

Diagramy aktivit modelují procesy jako kolekce aktivit a přechodů mezi nimi. Vzhledem k tomu, že diagramy aktivit jsou určeny zejména pro komunikaci s lidmi se znalostí struktury obchodních a podnikových procesů, měla by být dostatečně přehledné. [Kan2004]

6.2 Prvky diagramu aktivit

DEFINICE

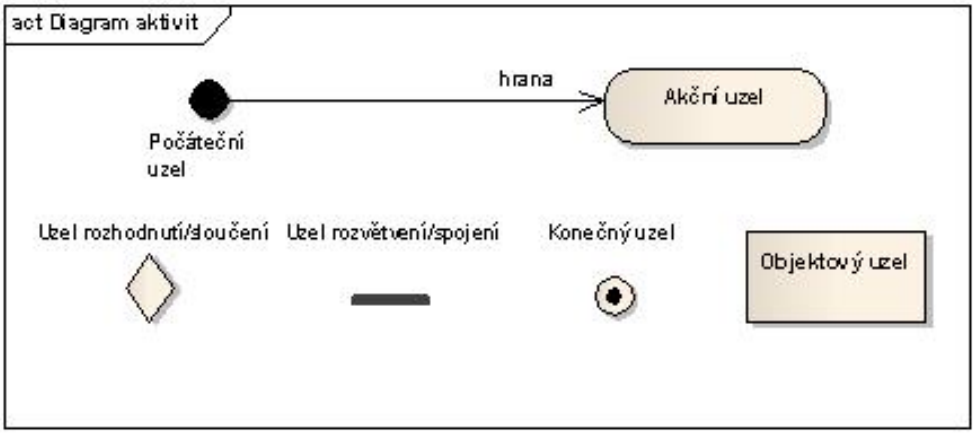
Diagram aktivit modeluje procesy jako aktivity, které se skládají z uzlů (nodes) vzájemně propojených hranami (edges), viz obrázek 12. [ARL2007]

Tři typy uzlů diagramu aktivit

Existují tři typy uzlů – akční uzly, které reprezentují samostatné a v rámci aktivity nedělitelné jednotky, řídící uzly, jejichž úkolem je řídit cestu uvnitř aktivity a uzly objektové, které zastupují objekty. Nejpoužívanějším akčním uzlem je tzv. call action node, který inicializuje aktivitu, chování či operaci. Příkladem řídících uzlů jsou počáteční (initial nodes), konečné uzly (final nodes) nebo uzly rozhodnutí (decision nodes). [Arl2008]

Uzel rozhodnutí má jednu vstupní hranu a několik konkurujících si hran výstupních. Daný výstup bude zvolen podle toho, která ze vzájemně se vylučujících kontrolních podmínek bude splněna. K označení hrany, která bude použita, pokud nebude splněna žádná kontrolní podmínka, se používá klíčové slovo jinak (else) [Fow2003], [Arl2008].

Uzel sloučení (merge) může mít několik vstupních, ale pouze jednu výstupní hranu. Používá se především pro sjednocení větví diagramu aktivit, které byly předtím rozděleny uzlem rozhodnutí [Arl2008].



Obrázek 12: Prvky diagramu aktivit, zdroj [ARL2007]

Procesy, které diagram popisuje, mohou probíhat paralelně, což je umožněno řídícími uzly rozvětvení (fork) a spojení (join) [Fow2003]. Pro zpřehlednění se může diagram rozdělit například dle rolí či organizačních jednotek do tzv. zón odpovědnosti či plaveckých drah (swimlanes) [Arl2008].

6.3 Doporučení k vytváření diagramu aktivit

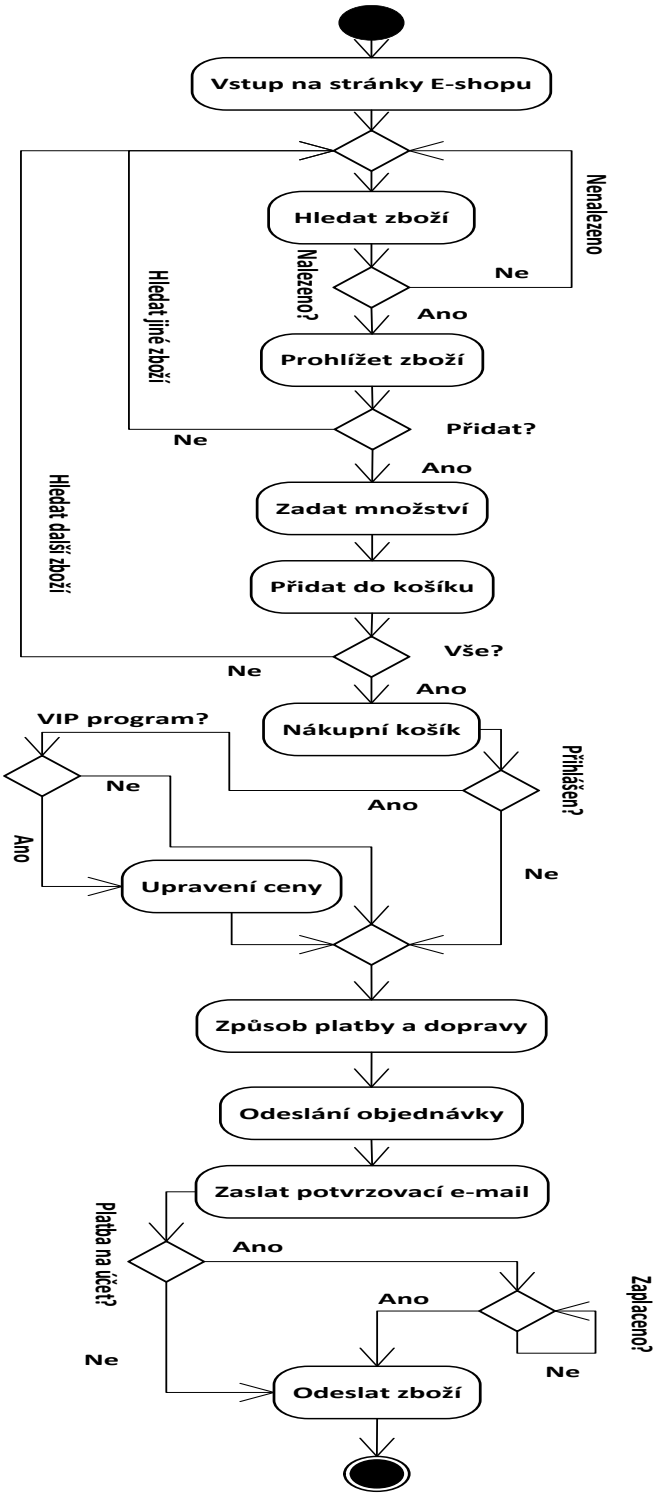
Podrobnější popis doporučení k vytváření stavového diagramu je zpracován v předchozím elektronickém skriptu [Fra2014], které je k dispozici studentům předmětu „Objektové metody modelování“ na e-learning portále SU v rámci připraveného kurzu.

SAMOSTATNÝ ÚKOL

Podle literatury [Fra2014], [Act2006] a případně i další literatury si sestavte pravidla pro vytváření diagramů aktivit. Tento návod aplikujte při kreslení diagramu aktivit ve své seminární práci.

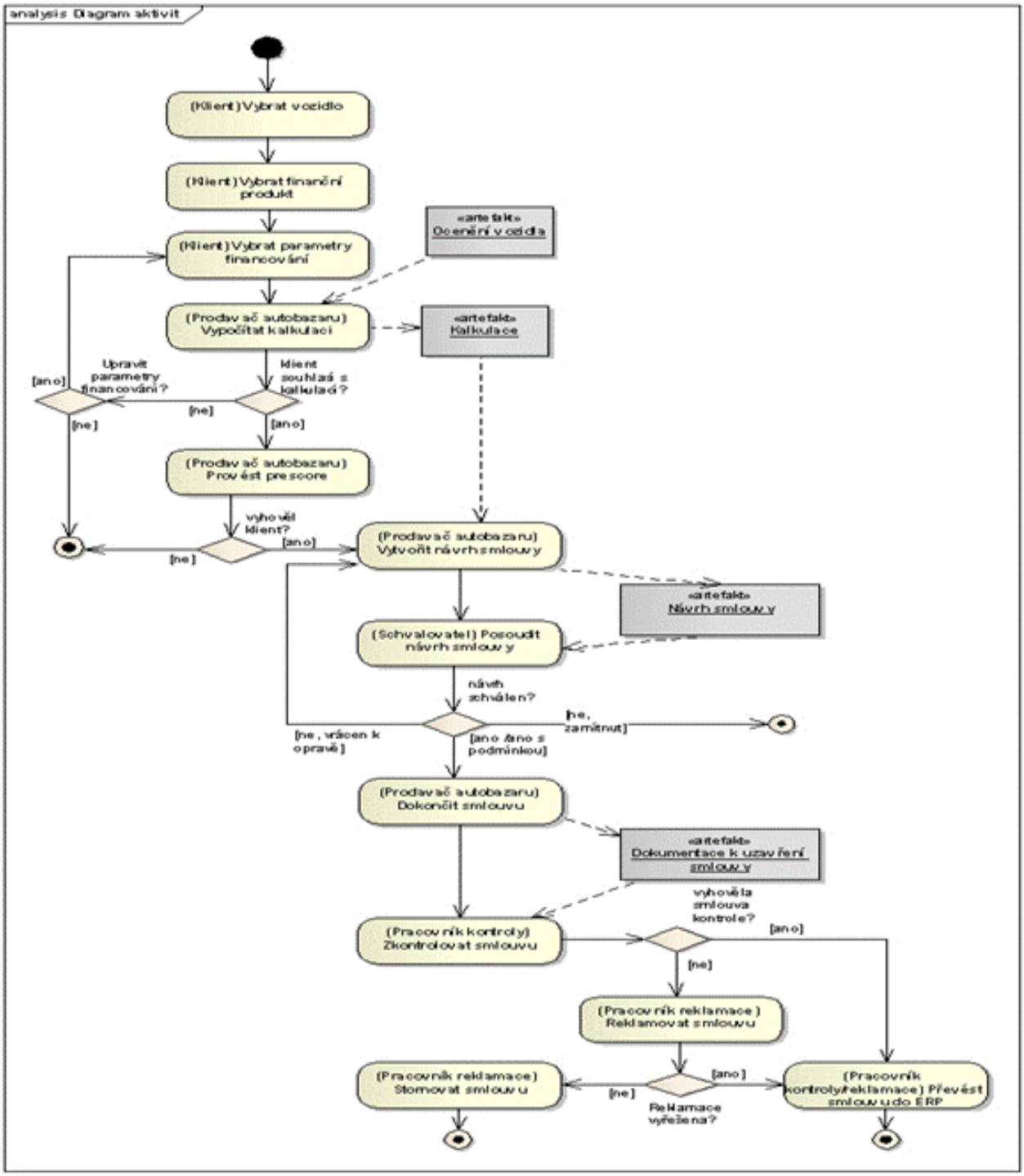
6.4 Příklady diagramů aktivit

Příkladový diagram v rámci případové studie znázorňuje chování systému e-shopu při tvorbě objednávky zákazníkem. Diagram prezentuje činnost zákazníka e-shopu od vstupu na stránky při tvorbě objednávky. Proces objednávání obsahuje vyhledání zboží, prohlížení popisu zboží a postupy při jeho zakoupení. Zákazník – uživatel e-shopu si zboží prohlíží, a pokud si ho vybere, přidá ho do nákupního košíku. Tento proces přidávání zboží opakuje uživatel tak dlouho, dokud nemá v košíku všechno zboží. Samozřejmě při procesu vkládání do košíku může volit množství, přednastavená hodnota počtu kusů je 1. Systém si před přechodem k dokončení objednávky zkontroluje, zda zákazník je přihlášen a má aktivovaný slevový VIP program. Pokud ano, tak systém upraví cenu. Následně je vyzván k vyplnění (v případě neregistrovaného zákazníka) nebo potvrzení (přihlášený zákazník) platebních, dopravních a doručovacích údajů. Následně je objednávka odeslána a systém zašle potvrzovací e-mail a na základě zvolené metody platby odešle zboží zákazníkovi.



Obrázek 13: Diagram aktivit - zákazník nakupuje na e-shopu, zdroj: vlastní zpracování

Druhým příkladem složitějšího diagramu aktivit je případ ocenění vozidla, viz obr. 14.



Obrázek 14: Diagram aktivit ocenění vozidla, zdroj: vlastní

SHRNUTÍ KAPITOLY

V této kapitole byla probrána metodika vytváření diagramu aktivit. Na příkladech byly vysvětleny základní pojmy a způsob využití diagramu aktivit.

OTÁZKY

Diagramy činností se používají pro vyjádření?

Vyberte jednu z nabízených možností:

a) Dynamického chování modelu.

b) Statické struktury modelu.

c) Vztahu mezi uživateli a systémem.

Které tvrzení o diagramech činností je pravdivé?

Vyberte jednu z nabízených možností:

a) Diagram činností není vybaven pro větvení dle podmínek a zachycuje jen paralelní zpracování.

b) Diagram činností je vybaven pro větvení dle podmínek a zachycuje paralelní zpracování.

c) Diagram činností je vybaven pro větvení dle podmínek, ale nezachycuje paralelní zpracování.

Diagramy činností patří mezi diagramy?

Vyberte jednu z nabízených možností:

a) Statické struktury systému.

b) Dynamického chování systému.

c) Žádná z uvedených dvou možností.

Diagramy činností podporují?

Vyberte jednu z nabízených možností:

a) Jen paralelní chování.

b) Jen sekvenční chování.

c) Jak sekvenční, tak i paralelní chování.

ODPOVĚDI

a. b. b. c.

7 SEKVENČNÍ DIAGRAM

RYCHLÝ NÁHLED KAPITOLY

Tato kapitola objasňuje princip sekvenčního diagramu.

CÍLE KAPITOLY

Cílem je porozumět a naučit vytvářet sekvenční diagram.

ČAS POTŘEBNÝ KE STUDIU

90 minut

KLÍČOVÁ SLOVA KAPITOLY

Sekvenční diagram, UML,

7.1 Základní charakteristika a prvky sekvenčního diagramu

Sekvenční diagram popisuje, jak spolu objekty komunikují v čase

Stavové diagramy pracují se stavy objektů. Popis stavů ovšem neřeší všechno. Proto v jazyku UML existuje další prostředek - diagram sekvencí, který popisuje, jak spolu objekty v čase komunikují. [Sch2001]

Sekvenční diagram nejčastěji zobrazuje chování a spolupráci jednotlivých objektů v rámci jednoho případu užití. Pro popis chování jednoho objektu napříč více případy užití se používá stavový diagram. [Fow2003]

Sekvenční diagram (Sequence Diagram) je nejvíce používaným diagramem interakcí. „Zachycuje grafický průběh zpracování v systému v podobě zasílání zpráv.“ [Buch2007]

PRVKY DIAGRAMU

Diagram sekvencí (sekvenční diagram) se skládá z objektů zakreslených běžným způsobem (jako obdélníčky s podtrženým jménem), ze zpráv zakreslených jako plné šipky a z času, který je znázorněn vertikálním postupem v diagramu. [Kan004] Zprávy mohou být v sekvenčním diagramu posílány jak mezi jednotlivými objekty, tak i třídami či dokonce aktéry. Proto se prvky, které mezi sebou v diagramu komunikují, nazývají souhrnně klasifikátory (classifiers). [Buch2007] Z každého klasifikátoru vede tzv. čára života (lifeline), která reprezentuje, jakým způsoben se instance určitého klasifikátoru účastní interakce. [Arl2008]

Diagram sekvencí se skládá z objektů, zpráv a znázornění času

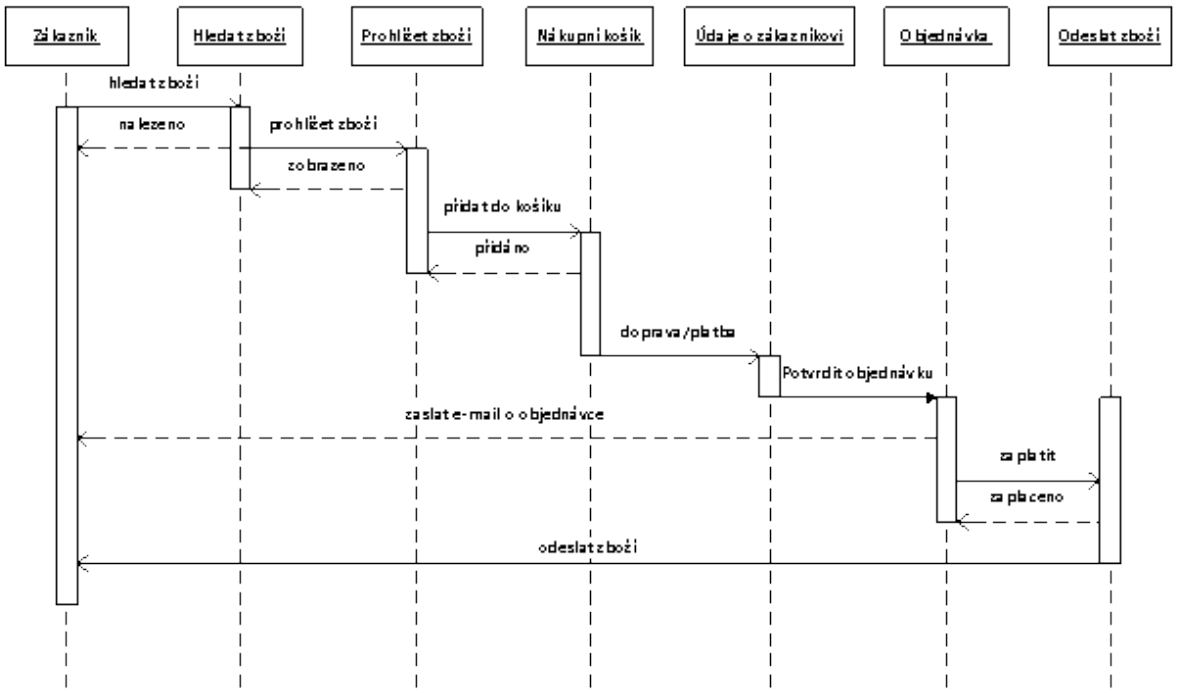
7.2 Doporučení k vytváření sekvenčního diagramu

Dle literatury [Seq2006], [Arl2008], [Buch2007] platí pro sekvenční diagram celá doporučení a návodů, například že aktéři by měli být pojmenování stejnými názvy jako v use-case diagramu. Podrobně byla pravidla vytváření sekvenčních diagramů zpracována v učebním textu [Fra2013], který je k dispozici na e-learning portálu v rámci kurzu „Objektové metody modelování“.

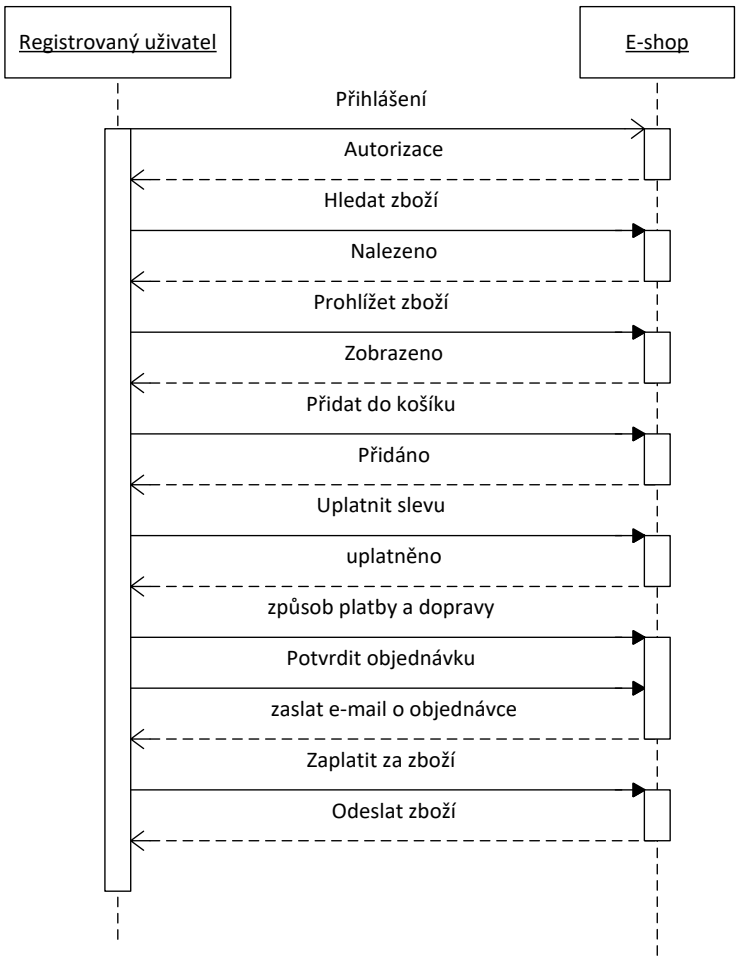
SAMOSTATNÝ ÚKOL

Podle literatury [Seq2006], [Arl2008], [Buch2007] a [Fra2014] si sestavte pravidla pro vytváření sekvenčního diagramu. Tato pravidla aplikujte při kreslení diagramu aktivit ve své seminární práci.

7.3 Příkladový sekvenční diagram



Obrázek 15: Sekvenční diagram popisující nákup zboží na e-shopu neregistrovaným uživatelem, zdroj: vlastní



Obrázek 16: Sekvenční diagram popisující nákup zboží na e-shopu registrovaným uživatelem s uplatněním slevy, zdroj: vlastní

SHRNUTÍ KAPITOLY

V této kapitole byla probrána metodika vytváření sekvenčního diagramu. Na příkladech byly ukázány základní prvky diagramu tak, aby studenti byli schopni za využití literatury sestavit sekvenční diagram.

OTÁZKY

Sekvenční diagram a diagram objektové spolupráce znázorňují?

Vyberte jednu z nabízených možností:

a) USE CASE - případ užití.

b) Statickou strukturu systému.

c) Jak spolu objekty spolupracují.

Jak se v sekvenčním diagramu vyznačuje zpráva?

Vyberte jednu z nabízených možností:

a) Šipkou s vyplněným hrotem.

b) Šipkou s polovinou hrotu.

c) Jednoduchou šipkou.

Jak se v sekvenčním diagramu zobrazuje čas?

Vyberte jednu z nabízených možností:

a) Vertikálně.

b) Nezobrazuje se.

c) Horizontálně zleva doprava.

Sekvenční diagram vystihuje tvrzení?

Vyberte jednu z nabízených možností:

a) Zobrazení funkcionality systému.

b) Zobrazení objektů v čase.

c) Zobrazení vzájemných vztahů objektů.

ODPOVĚDI

c. a. a. b.

---

8 PŘEHLED SOFTWARE PRODUKTŮ PRO PRÁCI

RYCHLÝ NÁHLED KAPITOLY

Kapitola dává přehled software, který se používá, resp. je dostupný na trhu. Jistě existuje celá řada software produktů, které umožňuji konstrukci UML diagramů. Zde jsou uvedeny dva základní produkt, které jsou využívány při výuce předmětu Objektové metody modelování.

CÍLE KAPITOLY

Cíl kapitoly dát přehled o nejčastěji používaných sw produktech a v případě SW Enterprise Architect dát podrobnější návod k jeho ovládání.

ČAS POTŘEBNÝ KE STUDIU

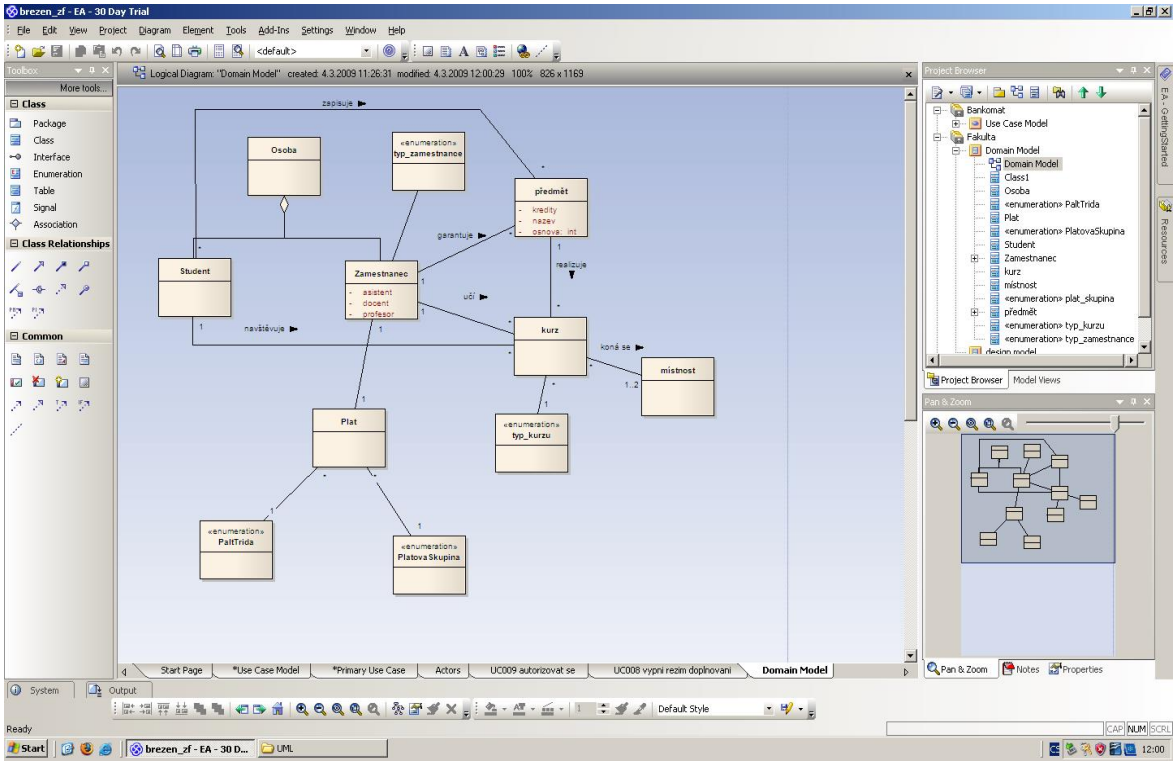
180 minut

KLÍČOVÁ SLOVA KAPITOLY

Software, UML diagramy, objektový orientovaný návrh SW.

8.1 ENTERPRISE ARCHITECT firmy SPARX

Velmi používaný program pro kreslení UML diagramů je Enterprise Architect firmy Sparx. Program je běžně využíván pro výuku pro transparentnost, přehlednost a poměrně jednoduché ovládání.



Obrázek 17: Vývojové prostředí Enterprise Architect firmy Sparx – CLASS DIAGRAM, zdroj: vlastní s využitím software Enterprise Architect

NÁVOD K OVLÁDÁNÍ PROGRAMU

EA firmy Sparx je ideálním CASE nástrojem, ve srovnání s Rational IBM Architect, je nepoměrně levnější a ve srovnání s šablonami na podporu UML v MS VISIO, to je komplexní CASE nástroj.

VYTVOŘENÍ PROJEKTU

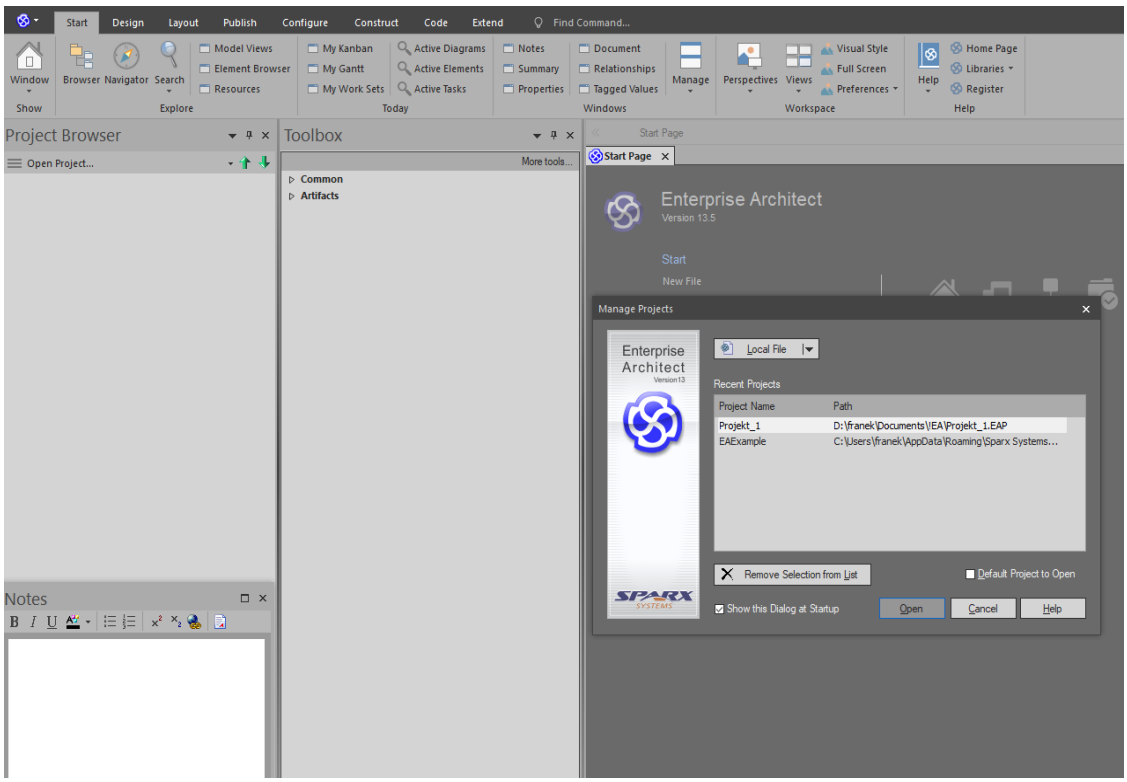
Projekt je tvořen jedním souborem nebo slouží jako úložiště pro jeden nebo více modelů. Prvním krokem je buď otevřít existující projekt, nebo vytvořit nový projekt. V tomto příkladu vytvoříme projekt založený na jednom souboru a přidáme některé modely založené na šablonách. Vytvoříme jednoduchý Use Case diagram, který budeme dále přizpůsobovat našim požadavkům. Kdykoliv můžeme znovu otevřít projekt, když na něj poklepneme v prohlížeči souborů. Objeví se také v našem seznamu Současné projekty (Recent Projects) na úvodní stránce (Start Page).

VYTVOŘENÍ NOVÉHO PROJEKTU:

1. Nastartujeme Enterprise Architect

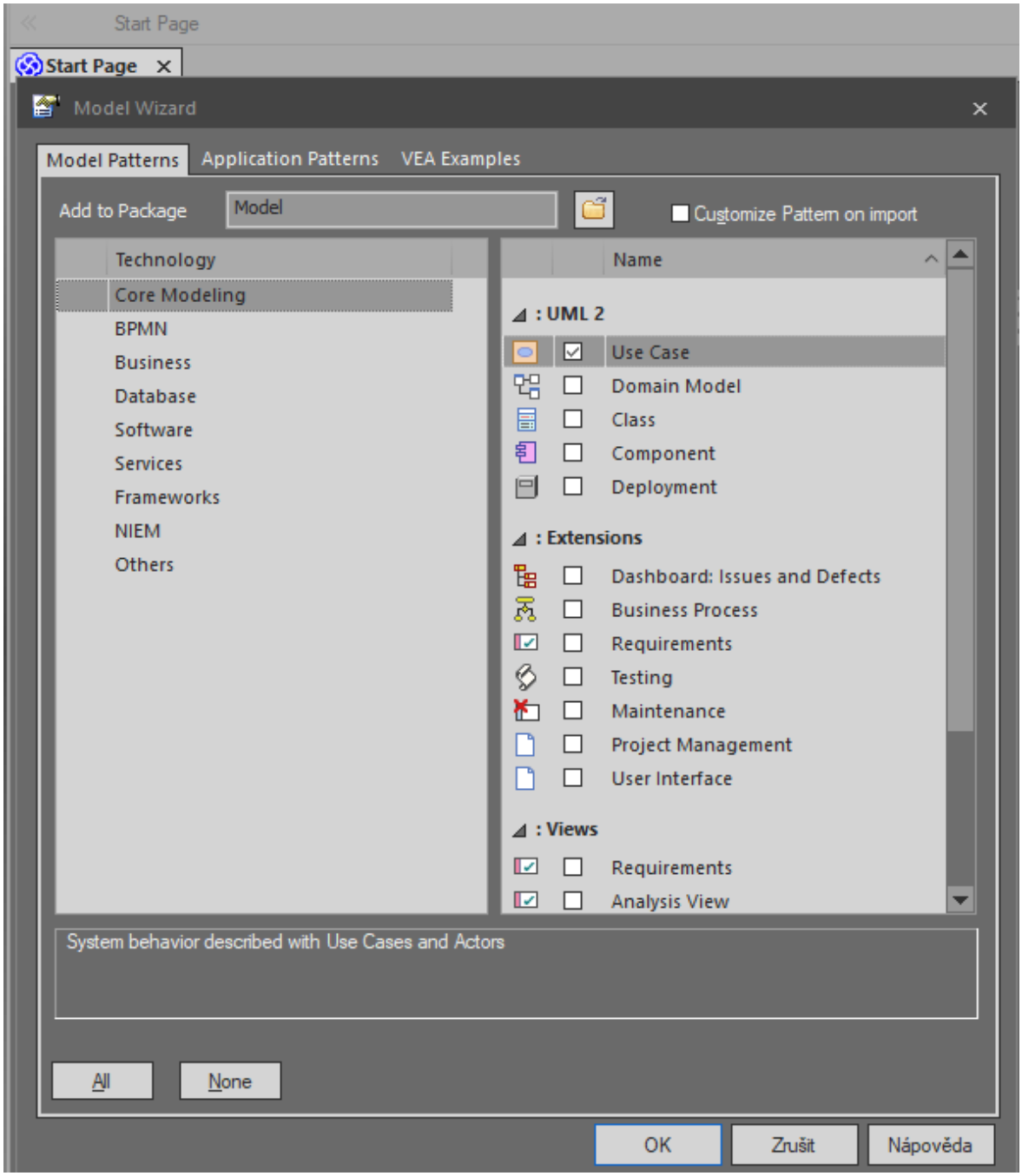
Zobrazí se pracovní prostředí EA a dialog „Manage Projects“ správy projektů se seznamem dříve použitých projektů, viz obrázek 18. Můžeme tak pokračovat v práci na již vytvořených projektech nebo si otevřít ukázkový projekt EAExample. Okno zavřeme.

2. Klikneme na tlačítko nový project (New File) a vybereme vhodné jméno a umístění nového projektu.



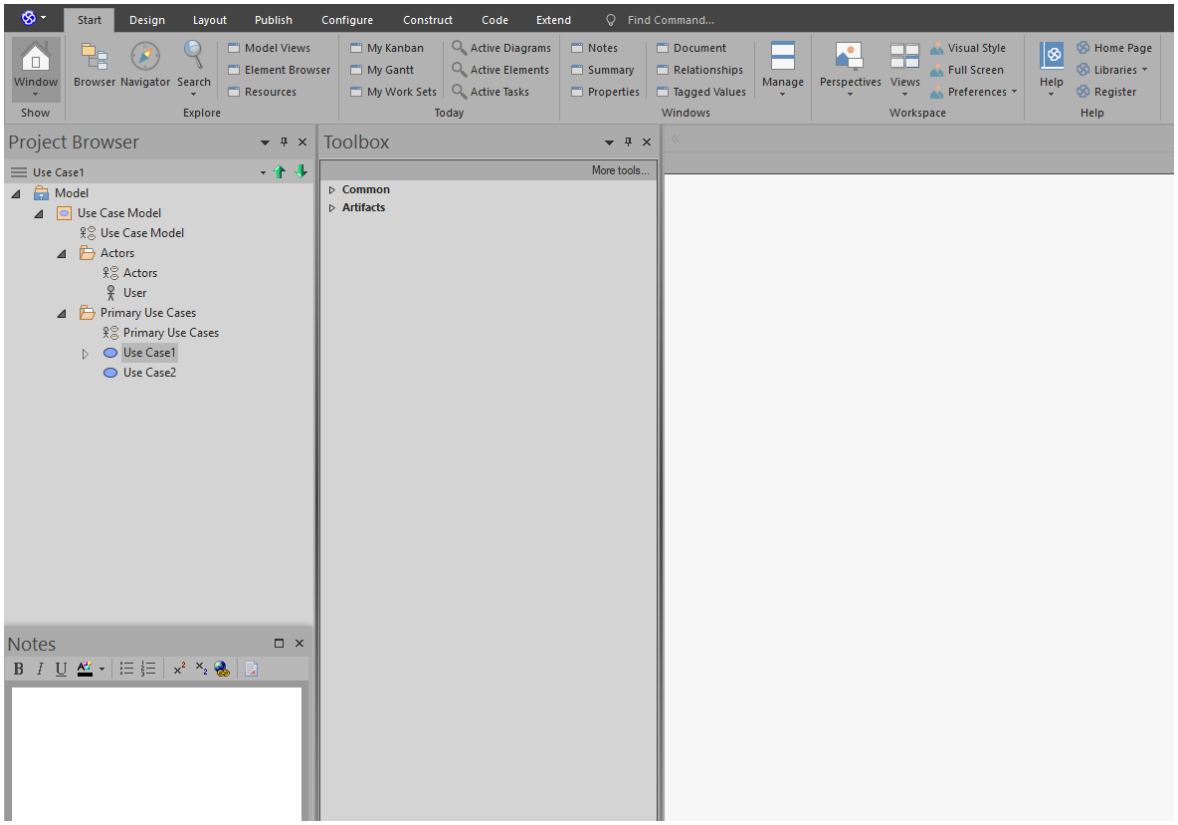
Obrázek 18: Vytvoření nového projektu v Enterprise Architect, zdroj vlastní

Otevře se standardní dialogové okno prohlížeče souborů Windows, soubory mají příponu .eap Projekt pojmenuje například newproject a soubor uložíme. Enterprise Architect vytvoří nový soubor projektu a uloží ho na určeném místě. Od této chvíle se celý projekt bude ukládat do souboru newproject.eap. Po uložení se nám otevře průvodce Model Wizard, kde zaškrtneme políčko Use Case, viz obrázek 19.



Obrázek 19: Model Wizard EA s volbou Use Case, zdroj vlastní

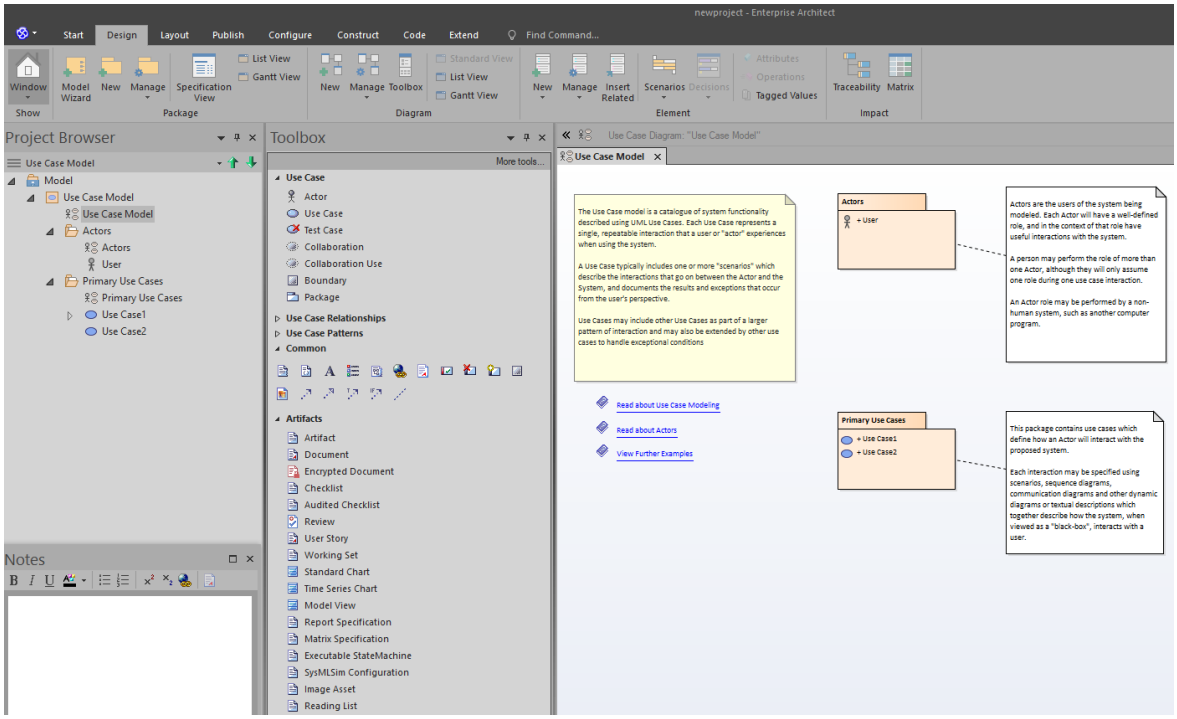
Průvodce (Model Wizard) automaticky vytvoří pro nás nový „Use Case Model“, který vidíme v Project Browser na levé straně vývojového prostředí, viz obrázek 20. Rozbalíme pomocí šipek model a dostáváme Use Case Model, Actors a Primary Use Cases.



Obrázek 20: Vytvoření Use Case Modelu, zdroj vlastní zpracování

NAKRESLENÍ USE CASE

Doubleclick na Use Case Model – otevře se nám ukázkový diagram Use Case, viz obrázek 21. Na obrázku je připraven důkladný návod a popis Use Case diagramu, balíčky Actors a Primary Use Case, viz též Project Browser.



Obrázek 21: Ukázkový Use Case Model, zdroj: vlastní zpracování

PŘIDÁNÍ PRVKŮ DIAGRAMU POMOCÍ PANELU NÁSTROJŮ

V podokně Toolbox máme připravené kreslící nástroje pro kreslení diagramu. Pokud Toolbox není vidět, aktivujeme ho kombinací kláves Alt+5. Metodou drag and drop přetáhneme ikonu Actor a dvakrát Use Case. Po přetažení se nám na pracovní ploše EA objeví příslušné obrázky Actor1 a Use Case1 a Use Case2. Při každém přetažení objektu se nám vysvítí příslušné okno s vlastnostmi. Ty mohu dle potřeby vyplnit, tedy pojmenovat Actory a Use a zejména u Use Case ve vlastnostech je textové pole pro zadání scénáře. Podobně přetažením myši vytvoříme asociaci mezi Actor1 a Use Case1 a Use Case2, viz obrázek 22.

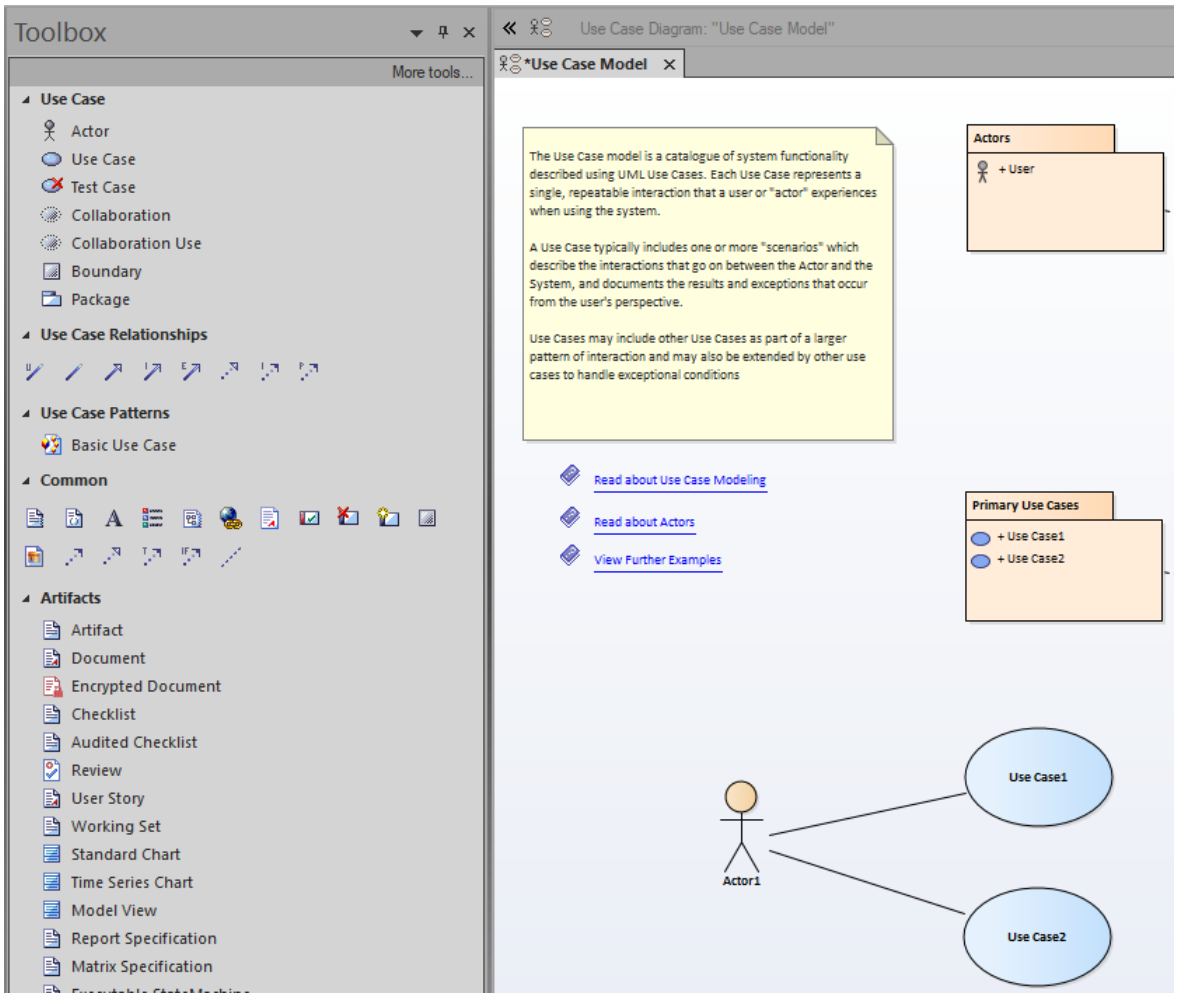
Pro vytvoření asociace můžeme využít i jiný postup. Po kliknutí na Actor1 se nám objeví pomocné navigační symboly, viz obrázek 23. Zatím jsme ponechali vzorové návody a balíčky na obrazovce. Ty můžeme jednoduše odstranit.

ODSTRANĚNÍ PRVKU

Odstranění prvku z pracovní plochy EA provedeme jednoduše tak, že prvek označíme myší a zmáčkneme klávesu DEL.

Prvek v Prohlížeči projektu můžeme mít ve více diagramech v rámci celého projektu. Pokud prvek odstraníme z diagramu pomocí klávesy Delete, nebude automaticky odstraněn odpovídající prvek v celém Prohlížeči projektu. Toho dosáhneme kombinací kláves CTRL + DEL.

Poznámka: Po stisknutí Ctrl + Delete budeme vyzváni k potvrzení odstranění a budeme varováni, že operaci nelze vrátit zpět.

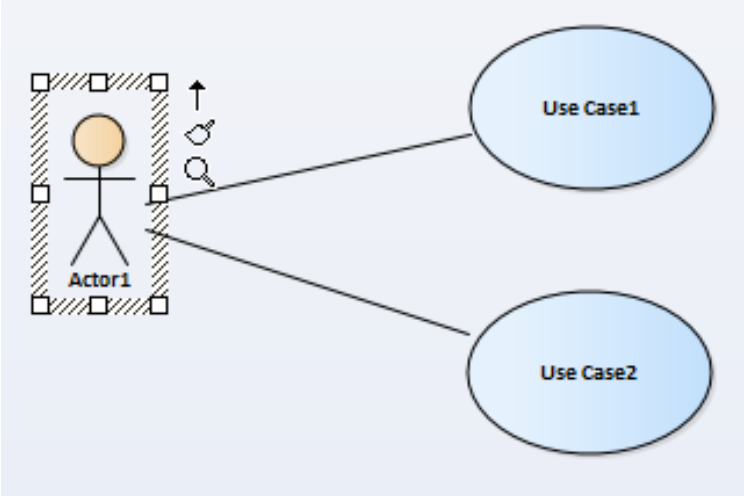


Obrázek 22: Kontextový Toolbox a přidávání prvků Use Case diagramu, zdroj: vlastní

POUŽITÍ QUICKLINKERU

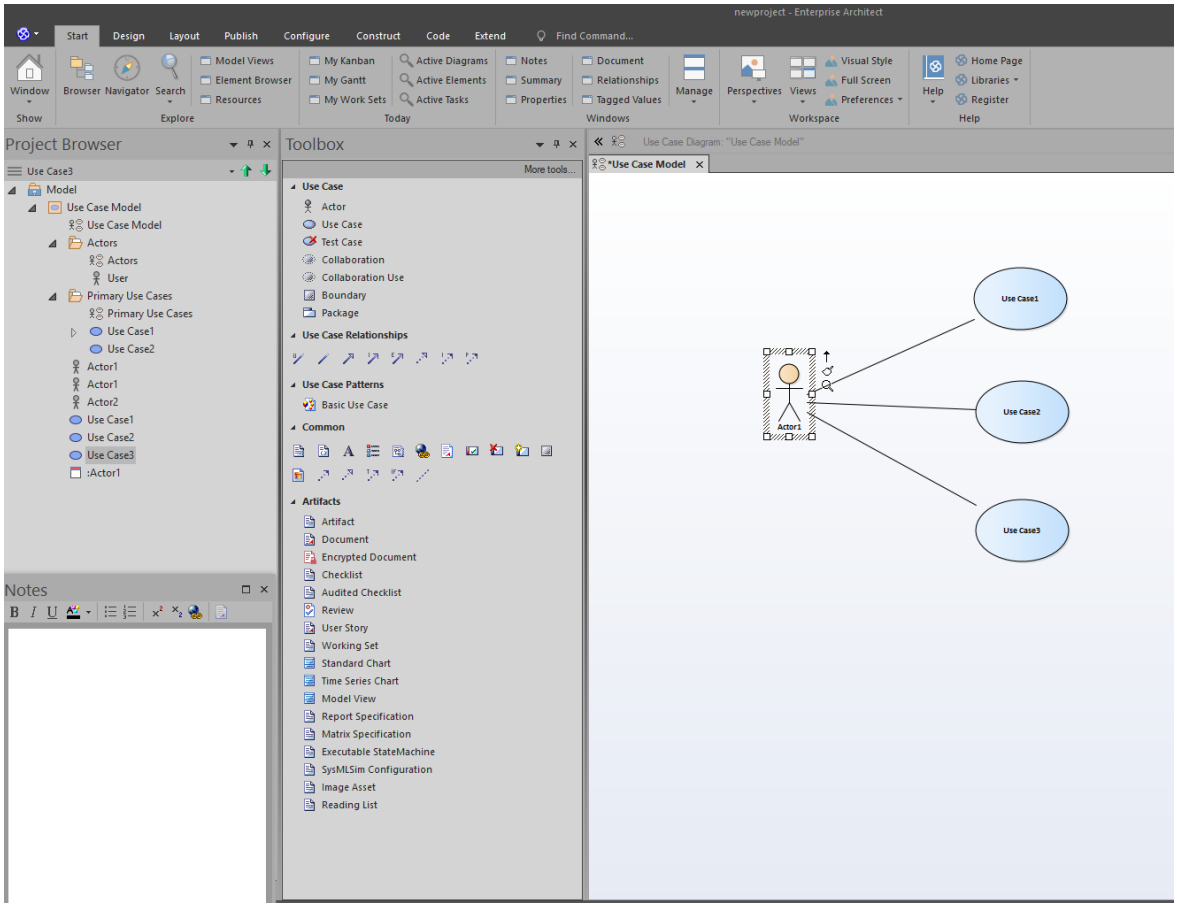
Použitím Quicklinkeru můžeme rychle vytvářet vztahy mezi stávajícími prvky bez použití Toolboxu. Po kliknutí na Actér1 se nám objeví příslušné konektory, viz obrázek 23.

Jednoduše přetáhneme šipku Quicklinkeru na požadované místo a pomocí menu vytvoříme prvek nebo konektor. Kontextové menu obsahuje nejběžnější prvky a konektory pro daný diagram.



Obrázek 23: Quicklinker – šipka, zdroj vlastní zpracování EA, zdroj: vlastní

V našem případě přidáme „Use Case 3“ a smažeme všechny návody a pomocné balíčky, viz obrázek 24.



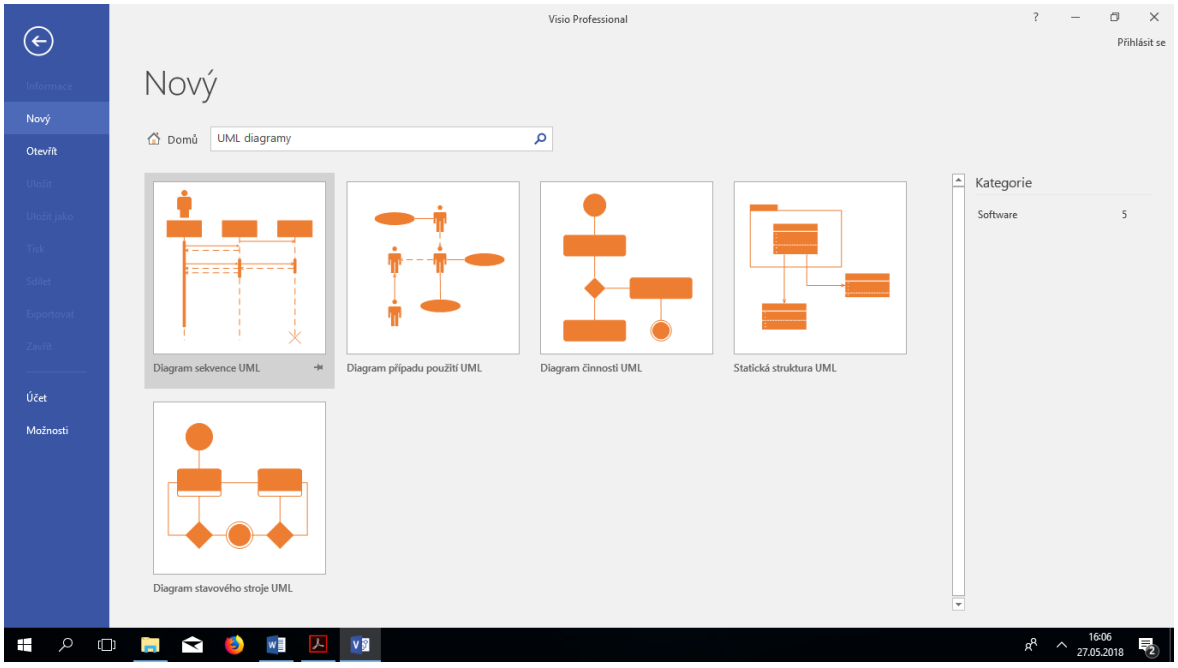
Obrázek 24: Vytvoření asociace pomocí Quicklinkeru, zdroj vlastní

8.2 UML A VISIO firmy MICROSOFT

Šablona diagramu modelu UML aplikace Microsoft Office Visio nabízí podporu vytváření objektově orientovaných modelů.

- Diagram případu použití
- Diagram statické struktury – diagram tříd
- Diagram činnosti
- Stavový diagram
- Sekvenční diagram
- Diagram komponent
- Diagram zavedení

Šablony MS VISIO pro UML vyhledáme zadání hesla UML diagramy do vyhledávání šablon a poté vybereme příslušný diagram, viz obrázek 25.



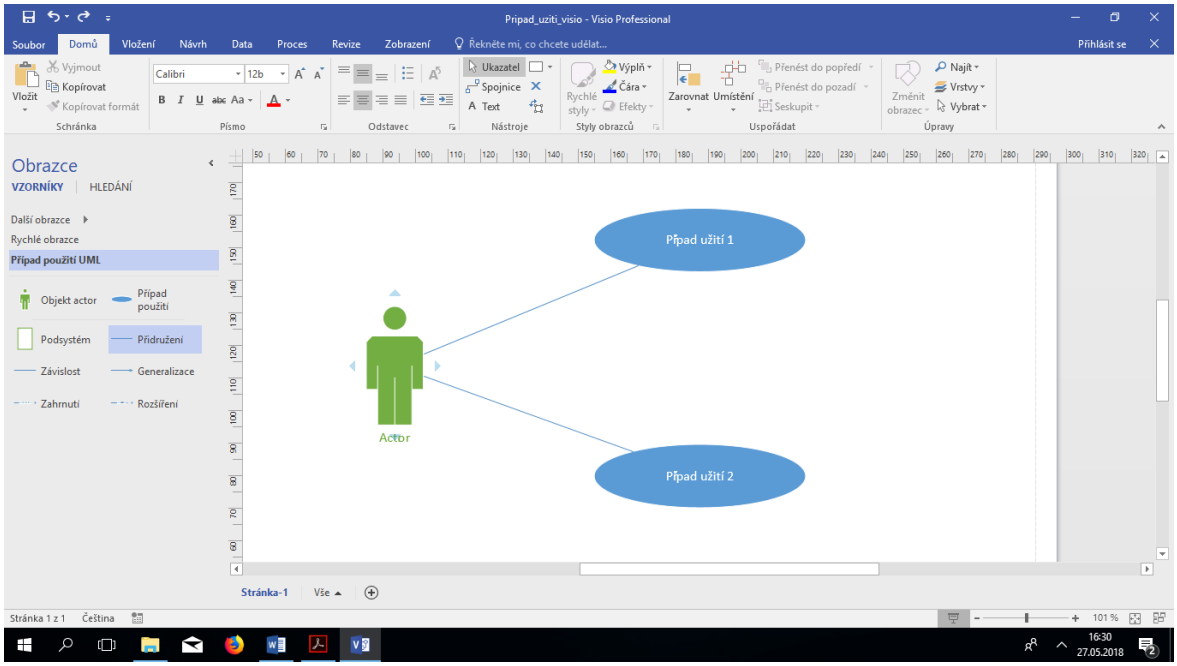
Obrázek 25: Vyhledání šablon na podporu UML diagramů, zdroj: vlastní v MS VISIO

Program VISIO je dostupný studentům Obchodně podnikatelské fakulty na počítačových učebnách, s šablonou diagramů UML na PC učebně, kde probíhá výuka předmětu „Objektové metody modelování.

Pro řešení seminární práce, kterou studenti dle sylabu předmětu „Objektové metody modelování“ musí vytvořit je podpora UML v MS VISIO postačující. Ideální prostředek pro řešení pokročilejších projektů jsou však výše uvedené CASE nástroje.

Návody na použití těchto šablon je nad možnosti rozsahu tohoto textu, ovládání je intuitivní, s využitím znalostí ovládacích postupů v MS OFFICE.

Prostředí a vzhled obrazovky šablony pro USE Case je na následujícím obrázku 26:



Obrázek 26: Printscreen prostředí kreslení USE CASE v MS VISIO, zdroj: vlastní zpracování

SHRNUTÍ KAPITOLY

V této kapitole je uveden přehled software pro vytváření diagramů UML a využití metodiky RUP. Zvláštní pozornost je věnována software ENTERPRISE ARCHITECT firmy Sparx. S tímto software počítáme do budoucna jako nejdůležitější a počítá se pořízením minimálně dvaceti plovoucích licencí. Software IBM Rational Architect je dostupný ve verzi z roku 2008 přes „tenký klient“ CITRIX. Tento software bude pouze demonstrativní s využitím postupů RUP.

Software VISIO obsahuje několik základních šablon pro kreslení UML diagramů. Je postačující pro psaní seminárních prací, pro psaní bakalářských a diplomových prací bude nutno použít EA.

SAMOSTATNÝ ÚKOL

1) Nainstalujte si software Enterprise Architect firmy Sparx, trial verzi na 10 dnů a vyzkoušejte si nakreslit diagramy uvedené v tomto skriptu.

2) Vyzkoušejte vzdálený přístup přes VM Ware software k programu Visio a vyzkoušejte si nakreslit diagramy v tomto skriptu

KONTROLNÍ OTÁZKA

Který, ze software Visio a Enterprise Architect firmy Sparx, umožňuje z diagramů generovat programový kód?

ODPOVĚDI

Enterprise Architect firmy Sparx.

9 RUP - RATIONAL UNIFIED PROCESS

RYCHLÝ NÁHLED KAPITOLY

Tato kapitola seznamuje s metodikou vytváření informačních systémů s názvem Rational Unified Process (dále RUP).

CÍLE KAPITOLY

Cílem kapitoly je vysvětlit metodiku RUP.

ČAS POTŘEBNÝ KE STUDIU

120 minut

KLÍČOVÁ SLOVA KAPITOLY

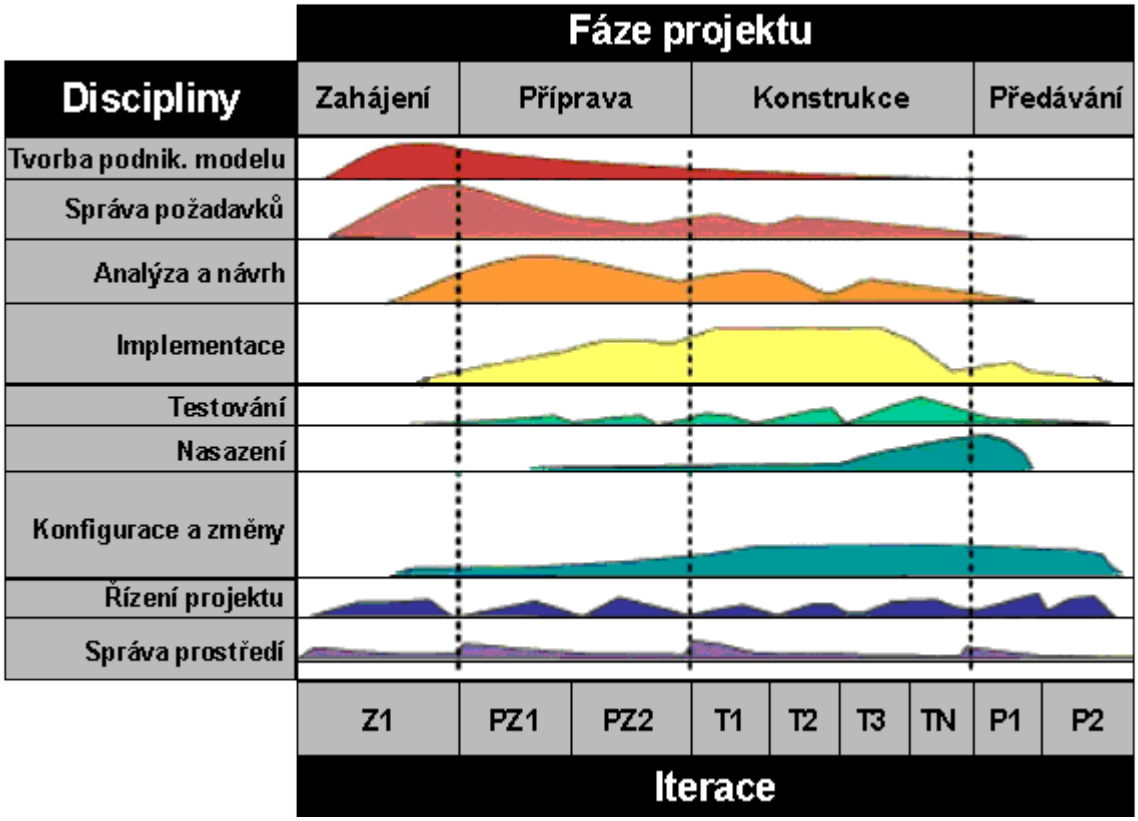
RUP, metodika, informační systém, vývoj sw, iterace, správa požadavků, komponenta, zahájeni, inception, příprava, elaboration, konstrukce, construction, předávání, transition

PRŮVODCE STUDIEM

Metodika RUP (komerční produkt) a UP (obecná metodika)

Metodiku Rational Unified Process Rup (RUP) vytvořila společnost „Rational Software Corporation“. Tato metodika byla distribuována formou softwarového produktu, který převzala i se společností v roce 2003 firma IBM. Nynější podobu metodiky a její realizaci pomocí software najdete na stránkách http//www.ibm.com - do vyhledání na stránkách nutno zadat slovo „rational“. Nutno poznamenat, že RUP je komerční produkt na rozdíl od obecné metodiky Unified Process.

Metodiku RUP názorně ilustruje grafické znázornění disciplín a fází projektu na obrázku 27:



Obrázek 27: Schéma projektu dle metodiky RUP, zdroj: [Arl2008]

Pro modelování procesů v životním cyklu projektu se využívá UML

Horizontální osa představuje čas, na vertikální ose jsou naneseny jednotlivé disciplíny, které se během projektu aplikují. Projekt je v této metodice rozdělen na čtyři fáze s danými cíli; jednotlivé fáze nejsou obvykle jednolité, nýbrž jsou prováděny v několika dílčích krocích (tzv. iteracích).

Pro modelování procesů v životním cyklu projektu – vývoje informačního systému se využívá diagramů jazyka Unified Modeling Language (UML).

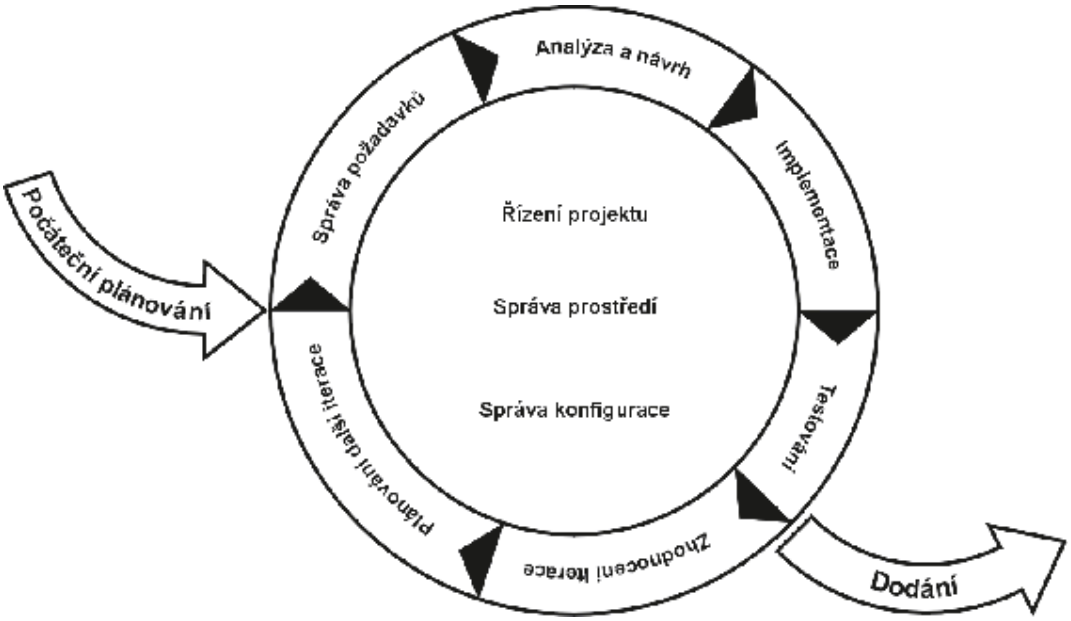
9.1 ZÁKLADNÍ PRAVIDLA RUP

Základní filosofií metodiky Rational Unified Process je šest základních pravidel, tzv. "nejlepších praktik" používaných při vývoji software:

1. Iterativní vývoj software
2. Správa požadavků
3. Architektura založená na komponentách
4. Vizuální modelování
5. Ověřování kvality software
6. Řízení změn software

ITERATIVNÍ VÝVOJ

U současných rozsáhlých systémů není již možné na začátku přesně definovat celý problém, navrhnout řešení, provést implementaci a až na závěr - kdy je spotřebována většina přidělených finančních prostředků - celý systém otestovat. Iterativní přístup spočívá v rozdělení celého projektu na čtyři fáze, z nichž každá se skládá z několika iterací se životním cyklem typu "vodopád". Výsledkem každé této iterace je spustitelná verze.



Obrázek 28: Iterativní vývoj, zdroj: ALDORF F., Metodika RUP

Tento přístup má zejména následující výhody:

- Možnost objektivního posouzení stavu projektu

- Rovnoměrnější pracovní vytížení vývojářského týmu. Přímo souvisí s předchozím bodem. Projekt je koncipován tak, aby co nejdříve přinášel konkrétní výsledky. Při rozdělení na iterace je možné snáze sledovat průběh projektu a dodržování stanovených termínů pro jednotlivé iterace.

- Možnost testování meziverzí

- Spolupráce s uživateli v průběhu celého projektu

- Včasné rozpoznání nesrovnalostí mezi požadavky, návrhem a implementací. Tato výhoda přímo vyplývá z předchozích dvou bodů. Díky tomu, že uživatelé mají možnost kontrolovat a hodnotit dílčí části systému, značně se omezuje riziko vysokých nákladů způsobených úpravami produktu v pozdní fázi vývoje.

- Snazší zapracování změn požadavků

AKTIVNÍ SPRÁVA POŽADAVKŮ

U vodopádového přístupu se požadavky sbírají a dokumentují pouze na začátku celého projektu, jejich změna v dalších stádiích vývoje se zpravidla nepřipouští. RUP naproti tomu využívá koncepci “aktivní správy požadavků” spočívající ve stálém kontaktu zadavatele s dodavatelem a možnosti zpřesňování a úprav požadavků v průběhu projektu.

KOMPONENTOVÁ ARCHITEKTURA

Komponenty v tomto kontextu označují netriviální části systému zajišťující určitou jeho funkcionalitu. Využití architektury systému založené na komponentách je výhodné z několika důvodů:

- Projekt lze snáze rozdělit mezi několik vývojářských týmů
- Zapracování změn je snazší
- Umožňuje postupnou tvorbu systému

Rational Unified Process podporuje standardní komponentové infrastruktury (COM, CORBA)

VIZUÁLNÍ MODELOVÁNÍ

Pro modelování se v RUP využívá Rational Unified Process prostředků jazyka UML (detailní popis viz standard [UML]).

Grafické znázornění systému a jeho chování zpřehledňuje návrh a umožňuje udržet konzistenci mezi modelem a vlastní implementací: např. software Rational Rose je schopen na základě modelu tříd vygenerovat kód v C++. Modul “C++ analyzer” pak umožňuje převést kód zpět na grafický model tříd.

OVĚŘOVÁNÍ KVALITY SOFTWARE

Softwarový produkt lze předat do provozu pouze v případě, že splňuje požadovaná kvalitativní kritéria:

- Funkcionalita: systém podporuje všechny funkce vymezené v modelu případů užití.
- Spolehlivost: systém správně řeší určenou skupinu chybových stavů.
- Výkon: dostupnost systému a doba odezvy jsou přijatelné.

V RUP je hlavním nástrojem pro zajištění výše uvedených parametrů testování. Testování jako jedna z disciplín této metodiky.

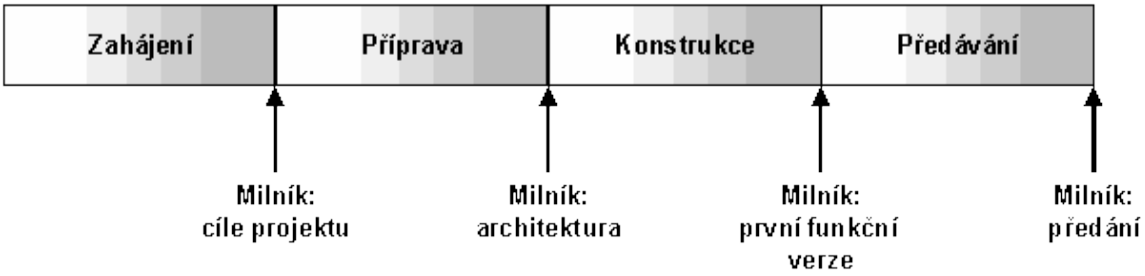
ŘÍZENÍ ZMĚN

Možnost provádět změny i v pozdějších stádiích projektu s sebou samozřejmě přináší i problém jejich řízení a dokumentace.

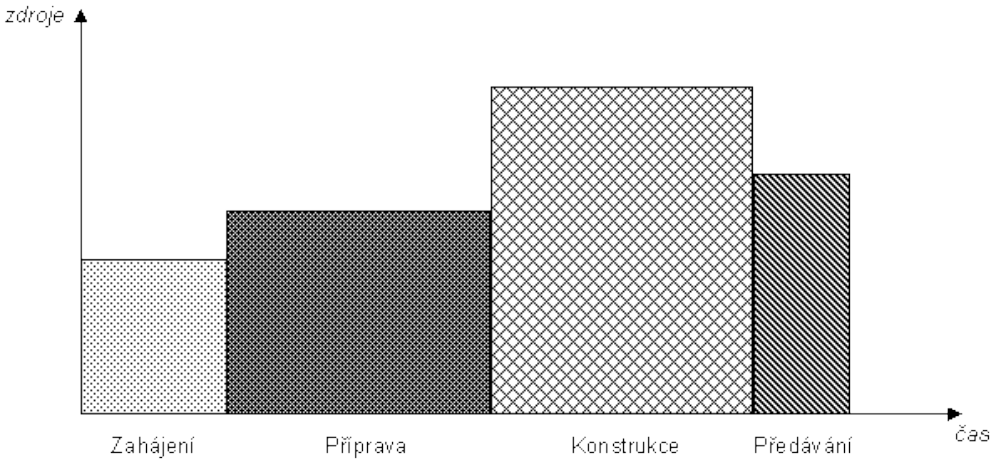
9.2 ŽIVOTNÍ CYKLUS PROJEKTU: CHARAKTERISTIKA FÁZÍ

ŽIVOTNÍ CYKLUS PRODUKTU V RUP

Životní cyklus projektu je v metodice Rational Unified Process rozdělen do čtyř základních fází (viz obrázek 29), z nichž každá je zakončena tzv. milníkem. Po dokončení každé fáze se provede zhodnocení, zda byly splněny požadované cíle. Další fázi projektu je možné zahájit pouze v případě splnění všech požadovaných kritérií.



Obrázek 29: Životní cyklus projektu, zdroj: ALDORF F., Metodika RUP



Obrázek 30: Obvyklá náročnost jednotlivých fází, zdroj:ALDORF F., Metodika RUP

Náročnost jednotlivých fází projektu na čas a zdroje závisí na typu vyvíjeného software, existenci nástrojů pro automatizaci vývoje a “generaci” software.

Vývoj produktu obvykle není definitivně ukončen uvedením první verze do provozu. Pokud software definitivně “neodumře”, vytvářejí se na základě nově vznikajících potřeb jeho další verze. Vývoj následující generace produktu se z pohledu řízení projektu skládá opět ze čtyř fází. U dalších vývojových cyklů se ale zpravidla podstatně zkracuje délka

prvních dvou fází, neboť analytické činnosti byly již z větší části provedeny u předchozích verzí.

9.3 ZAHÁJENÍ (INCEPTION)

Hlavním cílem této fáze je vymezení požadavků na systém, na němž se shodnou všechny zainteresované strany. Je třeba vymezit rozsah vytvářeného systému a pro klíčové činnosti vytvořit modely případů užití. U projektů, při nichž se pouze upravuje stávající systém, je tato fáze podstatně méně náročná, neboť výše uvedené kroky byly již dříve zčásti provedeny.

PLÁN ITERACE PRO FÁZI ZAHÁJENÍ:

1. Tvorba podnikového modelu: Základní charakteristika podniku

Analytik podnikových procesů popíše pomocí modelů případů užití procesy, které se dotýkají vytvářeného systému.

2. Správa požadavků: Analýza problémové oblasti

Na tomto kroku spolupracuje systémový analytik se zadavatelem. Jeho cílem je vymezit přesné určení systému a jeho rozsah. Rozhodujícím faktorem jsou požadavky uživatelů, již nyní je však třeba brát v úvahu omezení daná prostředím, do kterého budeme nový systém nasazovat (např. platformy, rozhraní na stávající systémy).

3. Správa požadavků: Identifikace požadavků zadavatele

Na základě konzultací se zadavatelem se sepíše seznam funkcí, které by systém měl podporovat. Doporučuje se průběžně vytvářet společný slovník používaných pojmů, který usnadní tvorbu modelů případů užití a udržení jejich konzistence. V tomto kroku je již možné sestavit úvodní model případů užití systému.

4. Řízení projektu: Plánování vývoje software

Tento krok představuje převedení vize do ekonomických dimenzí. Na základě požadavků na systém se sestaví úvodní plán vývoje software a počáteční odhad potřebných zdrojů pro další fáze (a samozřejmě s nimi souvisejících nákladů). V případě potřeby se doplní seznam rizik.

5. Správa prostředí: Příprava prostředí projektu

Zhodnocení stavu projektu i organizace zadavatele. Procesní inženýr vytvoří první verzi Konfigurace RUP. Specialista na vývojové nástroje zvolí prostředky pro analýzu požadavků. Systémový analytik vytvoří první verzi Pravidel pro modelování případů užití.

6. Řízení projektu: Plánování vývoje software

V tomto stádiu by již měla být jasně formulována vize systému a určena priorita mezi jednotlivými funkcemi systému. Vedoucí projektu na základě poslední verze seznamu případů užití (seřazeného podle priorit) a rizik aktualizuje plán vývoje software.

7. Řízení projektu: Ukončení a zhodnocení iterace

Rozsah prací, které je nutno provést před zahájením další fáze projektu, závisí především na tom, jak vedoucí projektu zhodnotí možné problémy (např. pokud je budován zcela nový systém, pracovníci dodavatele mají v této oblasti málo zkušeností apod.). Je-li projekt považován za málo rizikový, stačí pro dokončení jeho úvodní fáze pouze zpřesnit některé požadavky zadání. V opačném případě je třeba naplánovat ještě další iterace, při nichž se podrobněji zanalyzují problémové oblasti, vytvoří detailnější modely případů užití a dále rozpracuje úvodní návrh architektury.

VSTUPY

- Původní vize
- Stávající systém (pokud existuje)
- Úvodní požadavky na systém

VÝSTUPY

- Vize
- Výchozí verze ekonomického plánu projektu
- Plán vývoje software - včetně odhadu potřebných zdrojů. V této fázi se nemusí být údaj ještě příliš přesný, v dalších iteracích se bude postupně zpřesňovat.
- U složitějších projektů též úvodní návrh architektury - může se jednat o “nezávazný” prototyp, který se později přepracuje.

MILNÍK - CÍLE PROJEKTU

Na konci úvodní fáze projektu nastává důležitý milník, který rozhodne o jeho dalším pokračování. Pokud nebudou splněny následující podmínky, je třeba celý projekt přepracovat nebo v krajním případě i zastavit.

- Zadavatelé se musí shodnout s dodavateli na ceně a harmonogramu provádění prací
- Výstupní dokumenty musí prokázat, že dodavatel pochopil zadání
- Odhady rizik, nákladů a dob provádění musí být reálné

9.4 PŘÍPRAVA (ELABORATION)

Cílem této fáze je navrhnout stabilní model architektury systému, který poslouží jako základ pro jeho implementaci. Model architektury vychází ze základních požadavků na systém, vytváří se jeden nebo několik prototypů. Fáze přípravy navazuje na fázi zahájení projektu, na jejím začátku musí být k dispozici úvodní specifikace požadavků a úvodní verze plánu vývoje software zahrnující stručný plán iterací pro tuto fázi.

PLÁN ITERACE PRO FÁZI PŘÍPRAVY

1. Řízení projektu: Naplánování iterace

Plán iterace sestaví vedoucí projektu. Softwarový architekt doplní Architekturu software o popis kritérií pro její hodnocení, v případě zjištění možných problémů s architekturou aktualizuje Seznam rizik.

2. Správa prostředí: Příprava prostředí na iteraci

V tomto stádiu je třeba nakonfigurovat vývojové nástroje a vytvořit šablony dokumentů pro analýzu a návrh (tyto činnosti zajišťuje Specialista na vývojové nástroje a Procesní inženýr).

3. Správa požadavků: Identifikace klíčových případů užití, stanovení priorit

Softwarový architekt a vedoucí projektu určí, které případy užití se v této iteraci budou zpracovávat. Tyto případy užití budou klíčové pro návrh architektury. Vedoucí projektu následně doplní Plán iterace a provede se detailnější zpracování zvolených případů užití. Na základě nových informací pak softwarový architekt upraví model případů užití.

4. Analýza a návrh: Zpřesnění návrhu architektury

Na základě modelu případů užití, společného slovníku a zkušeností z praxe v daném oboru navrhne softwarový architekt způsob rozdělení na subsystémy a identifikuje klíčové třídy. V této fázi se rovněž zohlední prvky stávajícího systému, které se budou s novým systémem integrovat, včetně jejich rozhraní. Návrháři poté identifikují další třídy a provedou zpřesnění jejich definice (určí jejich atributy a vzájemné vazby). Třídy, které mají vliv na architekturu systému, zapracuje systémový architekt do dokumentu “Architektura systému”.

5. Analýza a návrh: Návrh subsystémů

Softwarový architekt zpřesní definici architektury pro potřeby zvoleného implementačního prostředí (programovací jazyk, relační databáze, distribuce systému aj.) a nadefinuje subsystémy pro návrh. Tyto jsou následně přiděleny jednotlivým návrhářům.

Za použití dříve nadefinovaných instancí tříd a subsystémů popisují návrháři realizaci zvolených případů užití (návrh komponent). V tomto kroku lze též provést identifikaci klíčových prvků datového modelu.

6. Implementace: Určení fyzické distribuce systému

Softwarový architekt v tomto kroku navrhne počáteční strukturu modelu implementace (tj. fyzické členění systému). Vstupem tohoto postupu jsou případy užití vyjádřené pomocí diagramů spolupráce.

7. Implementace: Plánování integrace systému

Systémový integrátor nyní určí, v jakém pořadí se budou jednotlivé subsystémy implementovat a později integrovat v rámci prototypu architektury (je třeba aktualizovat Plán vývoje software).

8. Testování: Určení předmětu testů

Vedoucí testování se dohodne s osobou odpovídající za příslušnou část systému, co se během této iterace bude testovat a navrhne způsob testování.

9. Implementace: Implementace komponent a Integrace subsystémů

Implementátoři nyní provedou vlastní implementaci tříd definovaných v předchozích krocích v rámci návrhu architektury. Systém je členěn hierarchicky, jednotlivé třídy jsou logicky seskupeny do modulů a subsystémů.

Jednotlivé části systému jsou poté samostatně odladěny a integrovány.

10. Implementace: Integrace systému

Systémový integrátor provede postupnou integraci vytvořených subsystémů do funkčního prototypu.

11. Testování: Ověření stability systému a Testování a hodnocení

V tomto kroku se provedou testy stávajícího systému (navrženy v kroku 8), výsledky se porovnají s požadovaným stavem. Systém testů se dle potřeby doplní pro potřeby dalšího testování.

12. Řízení projektu: Ukončení a zhodnocení iterace

Na závěr iterace provede vedoucí projektu její hodnocení na základě srovnání plánovaných výsledků, nákladů a termínů se skutečností. V případě potřeby se v tomto kroku naplánují úpravy systému v dalších iteracích. Vedoucí projektu sestaví/upřesní plán další iterace, provede aktualizaci Plánu vývoje software a Seznamu rizik.

Výsledkem úvodní iterace by měl být první funkční prototyp architektury, jednotlivé pohledy (případy užití, logický, procesní, implementační, pohled nasazení) by měly být celkem podrobně popsány.

Další iterace v této fázi slouží rozšíření modelu návrhu a implementace o další případy užití (na základě stanovených priorit). Počet těchto iterací závisí mj. na rozsahu projektu a zkušenostech dodavatele s danou oblastí.

VÝSTUPY

- Jeden či více prototypů architektury software.
- Model případů užití (nejméně z 80 % kompletní). Byli identifikováni všichni aktéři a všechny klíčové případy užití, k většině případů užití existuje popis.
- Model návrhu: proveden návrh alespoň pro 10 % případů užití.
- Datový model: byly identifikovány klíčové prvky datového modelu (tabulky, relace).
- Další požadavky netýkající se funkcionality a požadavky, které nelze přiřadit jednotlivým případům užití.
- Upravený seznam rizik a ekonomický plán.
- Hrubý plán projektu obsahující seznam iterací a kritéria hodnocení jednotlivých iterací.
- U systémů se složitým uživatelským rozhraním lze v této fázi vytvořit jeho prototyp.

MILNÍK - ARCHITEKTURA

Na konci fáze přípravy nastává druhý milník projektu. Projekt by v tomto stádiu měl splňovat následující podmínky:

- Vize produktu je již neměnná.
- Bylo identifikováno více než 80 % všech případů užití, alespoň polovina z tohoto počtu byla již analyzována.
- Návrh a implementace byly provedeny alespoň u 10 % případů užití.
- Návrh architektury je již konečný, nebude docházet k zásadním změnám.
- Jsou definovány postupy pro hodnocení a testování.
- Testování prototypů prokázalo, že byly správně identifikovány a podchyceny rizikové oblasti.
- Plán iterací pro fázi konstrukce je dostatečně podrobný a kvalitní, aby bylo možné v projektu pokračovat. Tyto plány jsou podpořeny důvěryhodnými odhady dalšího vývoje.

- Zadavatelé souhlasí s realizací software s použitím navrhované architektury a plánu vývoje; shodují se na tom, že takový software splní požadavky vytyčené ve vizi.
- Skutečná spotřeba zdrojů nepřevyšuje plán.

9.5 KONSTRUKCE (CONSTRUCTION)

Hlavním cílem této fáze je dokončení návrhu, vytvoření a otestování první funkční verze systému. Na rozdíl od předchozích dvou fází je tato zaměřena převážně na implementaci. Základním problémem je efektivní řízení zdrojů - stejně jako v předchozích fázích je vhodné projekt rozdělit na relativně nezávislé části a ty pak přidělit jednotlivým vývojářským týmům. V této fázi projektu by měly být požadavky na systém již stabilní, správa požadavků by měla řešit pouze případné omezuje pouze na zapracování dodatečných požadavků na změny.

PLÁN ITERACE PRO FÁZI KONSTRUKCE

1. Řízení projektu: Naplánování iterace

Vedoucí projektu naplánuje, jaká funkcionalita se během této iterace bude vytvářet.

2. Prostředí: Příprava prostředí na iteraci

Procesní inženýr dále zpřesní Konfiguraci RUP, šablony dokumentů a pravidla vývoje.

3. Implementace: Plánování integrace systému

Plán integrace stanovuje pořadí, ve kterém se jednotlivé moduly systému budou přidávat do testovací konfigurace. Tento plán sestavuje systémový integrátor.

4. Testování: Tvorba testů

Návrhář testů vytvoří strukturu testů pro ověření jednotlivých požadavků na systém. Pro každý požadavek, který lze testovat, by měl být navržen alespoň jeden test (lze použít i některé upravené testy z předchozích iterací).

5. Analýza a návrh: Návrh subsystémů

Návrháři zpřesní definici vybraných tříd/modulů určených v předchozích iteracích (např. doplní některé atributy).

6. Implementace: Implementace komponent

Implementátoři naprogramují jednotlivé komponenty a provedou jejich otestování. Tyto základní testy by měly ověřit správnost implementace a korektní chování komponenty při vstupu správných i chybných dat.

Během implementace se mohou objevit nové skutečnosti, na jejichž základě se provede úprava modelu návrhu.

7. Implementace: Integrace subsystémů

Z komponent vytvořených více implementátory se na základě plánu integrace sestaví subsystém.

8. Testování: Tvorba testů

Návrháři a implementátoři testů vytvoří testy pro ověření integrace subsystémů i celého systému. Tyto testy mají za úkol ověřit, zda jednotlivé komponenty správně spolupracují prostřednictvím použitých rozhraní.

9. Testování: Testování a hodnocení

V tomto kroku se na příslušný subsystém aplikují testy vytvořené v krocích 4 a 6. Pokud jsou při testování zjištěny chyby, sestaví testeři hlášení o chybě (automaticky znamená i požadavek na změnu).

10. Implementace: Integrace systému

Po sestavení a otestování subsystému je tento poskytnut integrátorovi pro začlenění do celého systému. \\ Integrátor postupně přidává jednotlivé subsystémy a sestavuje testovací verzi produktu (u počátečních verzí se testování provádí po přidání každého subsystému).

11. Testování: Testování a hodnocení

Na celý systém se aplikují testy integrace vytvořené v kroku 6. Pokud jsou při testování zjištěny chyby, sestaví testeři hlášení o chybě.

12. Testování: Testování a hodnocení

Cílem tohoto kroku je otestovat funkčnost celého systému. Na produkt se aplikují testy vytvořené v krocích 4 a 6. Pokud jsou při testování zjištěny chyby, sestaví testeři hlášení o chybě.

13. Řízení projektu: Ukončení a zhodnocení iterace

Vedoucí projektu spolu se specialistou na vývojové nástroje a procesním inženýrem zhodnotí iteraci z pohledu nákladů, dodržování harmonogramu a využití vývojových nástrojů.

Další iterace ve fázi konstrukce

Další iterace slouží převážně k návrhu a implementaci další funkcionality systému. Ke konci této fáze je též kladen důraz na testování a případně i plánování nasazení systému.

VÝSTUPY

- Software: První funkční verze vyvíjeného systému.
- Plán nasazení: První verze.
- Model implementace: soubor komponent vytvořených v této fázi.
- Model testování: soubor testů vyvinutý pro ověřování verzí systému vyvinutých ve fázi konstrukce.
- Uživatelská dokumentace: první verze manuálů; je třeba hlavně u systémů s rozsáhlým uživatelským rozhraním.
- Model návrhu: po dokončení této fáze by měly být identifikovány již všechny požadavky a zapracovány do modelu návrhu.
- Plán iterací pro fázi předávání.
- Datový model: úplná specifikace datového modelu (tabulky, relace, indexy…).

MILNÍK - PRVNÍ FUNKČNÍ VERZE

Po dokončení fáze konstrukce by měl být systém připraven k předání. Kromě "alfa" či "beta" verze software musí existovat i uživatelský manuál včetně aktuální dokumentace. Projekt by v tomto okamžiku měl splňovat následující podmínky:

- Produkt je dostatečně stabilní, aby je bylo možné poskytnout uživatelům.
- Zadavatel je připraven k nasazení systému.

Pokud produkt některé tyto podmínky nesplní, je vhodné odložit předání o jednu verzi.

9.6 PŘEDÁVÁNÍ (TRANSITION)

Hlavním cílem této fáze je předání finální verze systému koncovým uživatelům. V jejím průběhu se provádí beta testování systému a jeho drobné úpravy na základě připomínek uživatelů. V této fázi by již nemělo docházet k žádným zásadním změnám funkcionality software - požadavky zadavatele by se měly týkat pouze instalace, konfigurace a odlaďování drobných chyb zjištěných při testovacím provozu.

PLÁN ITERACE PRO FÁZI PŘEDÁVÁNÍ

1. Řízení projektu: Naplánování iterace

Hlavním cílem této fáze je předání software splňujícího požadavky na spolehlivost, výkon a funkcionalitu. Vedoucí projektu nyní rozhoduje o dalším vývoji hlavně na základě

požadavků na změny (odstraňování závad a připomínky uživatelů k testovací verzi). Vedoucí projektu odpovídá též za plánování podpory pro koncové uživatele, zajištění materiálu potřebného pro instalaci produktu a přípravy na přejímací testy.

2. Nasazení: Beta testování

Před vlastním oficiálním předáním produktu je vhodné dát uživatelům možnost otestovat jeho beta verzi a verzi určenou pro předání doladit na základě jejich připomínek.

3. Analýza a návrh, Správa požadavků

V této fázi se již předpokládá, že požadavky budou téměř neměnné, nebude docházet k ad hoc úpravám produktu. Mohou se ale vyskytnout změny, které bude třeba do systému ještě zapracovat. Analýza a návrh zahrnují v této fázi hlavně drobné úpravy architektury na základě testování - ladění a změny fyzické distribuce komponent systému.

4. Implementace

Hlavním činností v rámci implementace je odstraňování závad zjištěných při provozu beta verze sytému a zapracování případných požadavků na změny. V tomto stádiu by se již neměly vytvářet zcela nové komponenty. Stejně jako v předchozí fázi se po provedení změn provedou testy funkčnosti jednotlivých komponent.

5. Testování

Stejně jako u předchozího kroku spočívá testování systému převážně v ověřování správnosti provedených změn. Obvykle ale není třeba zevrubně testovat integraci jednotlivých komponent, neboť jejich rozhraní jsou v tomto stádiu již stabilní. Návrh testů se omezuje na jejich přizpůsobení upraveným verzím komponent. U řady projektů jsou též smluvně upraveny přejímací testy - jedná se o otestování předávané verze systému pomocí stávající série testů.

6. Nasazení: Přejímací testy

Zadavatel provede před převzetím produktu jeho formální otestování. Zpravidla se použije systém testů vyvinutý v rámci testování pracovních verzí u dodavatele.

7. Výroba softwarového balíku / Poskytnutí přístupových práv ke stažení SW

Na předání systému musí být připraveny obě smluvní strany. Dodavatel proto musí zajistit následující:

- Vlastní dodání systému: vedoucí nasazení naplánuje vytvoření a předání instalací software (instalační programy vytvoří implementátor) a dalšího materiálu potřebného pro uvedení systému do “ostrého” provozu.

- Vytvoření a dodání konečné verze uživatelské dokumentace: Autor uživatelských manuálů vytvoří jejich konečné verze.
- Školení pracovníků zadavatele: Autor podkladů pro školení vytvoří poslední verze materiálů, podle nichž se bude provádět školení uživatelů.

VÝSTUPY

- Software: konečná funkční verze splňující všechny požadavky zadání, jsou k dispozici instalační programy a média.
- Manuály: uživatelská dokumentace k aktuální verzi produktu.
- Materiály pro školení uživatelů k aktuální verzi produktu.

MILNÍK - PŘEDÁNÍ

Tímto milníkem končí vývoj stávající verze software, produkt je předán zadavateli a přechází do ostrého provozu. Základními kritérii hodnocení úspěšnosti projektu jsou spotřeba zdrojů a spokojenost uživatelů.

Pro dodavatele software začíná nyní stádium poinstalační podpory - poté, co je systém nasazen do rutinního provozu a podrobně se s ním seznámí větší množství uživatelů, obdrží zadavatel zpravidla řadu návrhů na jeho zlepšení.

V závislosti na závažnosti připomínek se nyní rozhodne, zda se zahájí nový projekt a vytvoří další verze celého software či zda bude pouze stačit provést úpravy některých komponent.

SHRNUTÍ KAPITOLY

Tato kapitola je věnována popisu metodiky tvorby informačních systémů. Samotné UML se svými diagramy nejsou metodikou. Ty metodice pomáhají při tvorbě software. Veškeré studijní materiály o metodice RUP a srovnání s UP jsou obsaženy v [FRA2014].

OTÁZKY

Zkratka UP znamená?

Vyberte jednu z nabízených možností:

a) Unified process.

b) Unified program.

c) Universal process.

Co je UP?

Vyberte jednu z nabízených možností:

a) Soubor typových řešení iterativní metodologie vývoje SW, resp. IS.

b) Objektově orientovaná iterativní metodologie vývoje SW, resp. IS.

c) Jiný název pro UML diagramy.

Axiomem metodiky UP není?

Vyberte jednu z nabízených možností:

a) Zásada řízení případem užití a rizikem.

b) Zásada soustředění na programování.

c) Zásada iterace a přírůstku.

Pět základních pracovních postupů (workflow) je?

Vyberte jednu z nabízených možností:

a) Plánování, Analýza, Návrh, Implementace, Testování.

b) Analýza, Návrh, Programování, Implementace a Testování.

c) Požadavky, Analýza, Návrh, Implementace a Testování.

Fáze metodiky UP jsou?

Vyberte jednu z nabízených možností:

a) Milník, Fáze, Iterace a Zavedení.

b) Zahájení, Konstrukce, Zavedení a Testování.

c) Zahájení, Rozpracování, Konstrukce a Zavedení.

Software je v metodice UP vytvářen v iteracích.

Vyberte jednu z nabízených možností, která charakterizuje iteraci v rámci UP:

a) Každá iterace je mini projekt.

b) Iterace jsou skládány jedna za druhou, ale nemají vliv na konečnou podobu systému.

c) Iterace jsou aplikovány jen na celý projekt.

Co generuje každá iterace?

Vyberte jednu z nabízených možností:

a) Baseline.

b) Přírůstek programového kódu.

c) Helpline.

Tvrzení "Pro každý nový projekt tvorby SW je třeba vytvořit novou instanci metodiky UP" je pravdivé?

Vyberte jednu z nabízených možností:

a) ANO.

b) Někdy ANO někdy NE - záleží na použití metodiky UP.

c) NE.

Metodika UP a RUP se liší?

Vyberte jednu z nabízených možností:

a) Sémantikou elementů jednotlivých metod.

b) Jsou to úplně odlišné metodiky.

c) RUP vykazuje určité terminologické a syntaktické odlišnosti.

Metodika UP a RUP je rigorózní nebo agilní?

Vyberte jednu z nabízených možností:

a) Agilní i rigorózní.

b) Agilní.

c) Rigorózní.

ODPOVĚDI

a. b. b. c. c. a. a. a. c. c.

10 PRAKTICKÉ PŘÍKLADY VYUŽITÍ UML

PŘÍPADOVÉ STUDIE

Tato kapitola přináší případové studie – čerpající příklady ze seminárních prací, které vznikly při výuce předmětu „Úvod do objektového modelování“. Kapitola je zařazena proto, aby studenti inovovaného předmětu „Metody objektového modelování“ měli inspiraci při tvorbě seminárních prací. Zde uvedené řešené příklady – případové studie poslouží pro praktické procvičení uplatnění UML diagramů při návrhu IS s využitím software MS VISIO a Enterprise Architect firmy Sparx.

10.1 Skladový informační systém

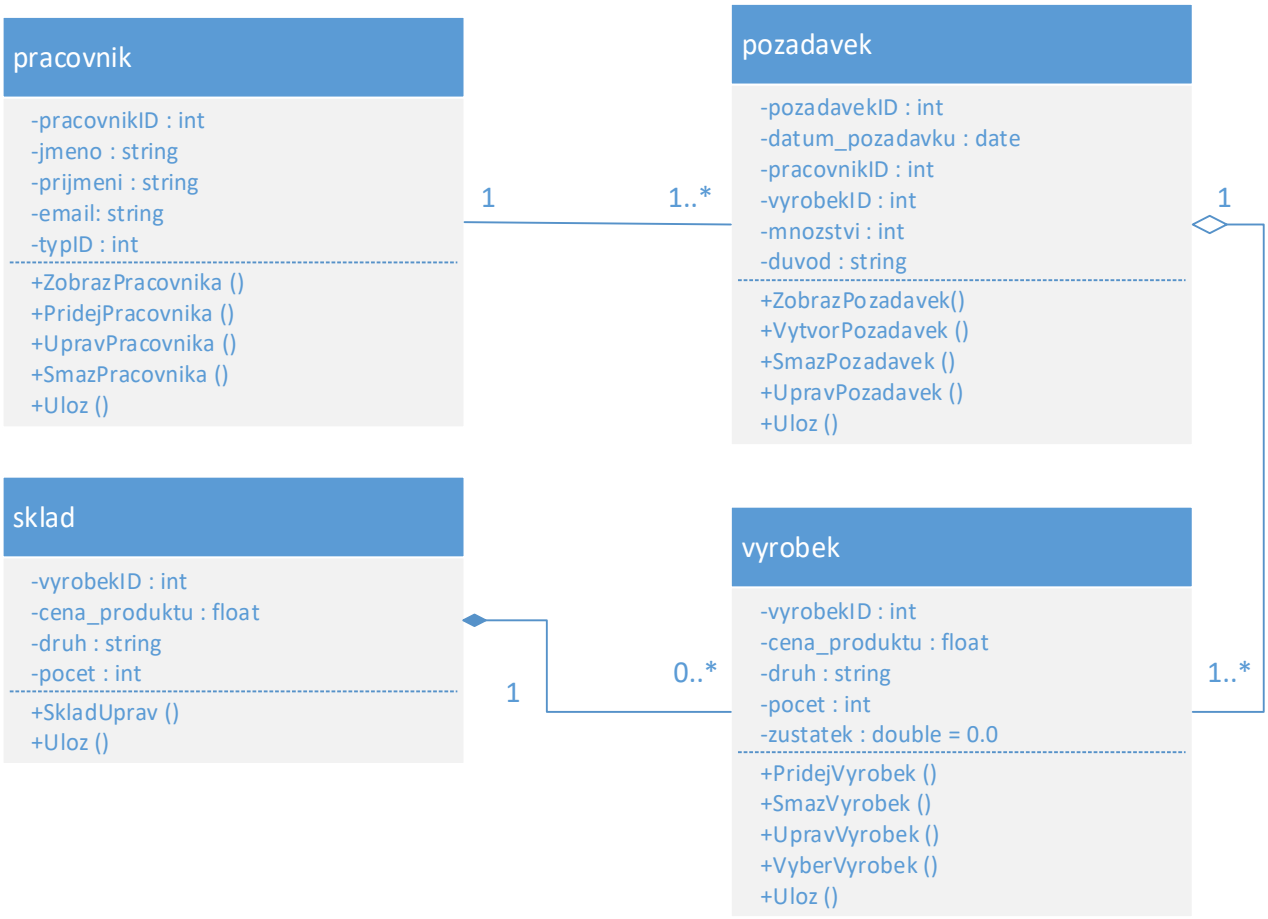
Úvod

Tématem případové studie je navrhnout modul informačního systému pro sklad. Základní požadavky jsou, aby mohl jakýkoliv libovolný pracovník pomocí IS zažádat o vyskladnění, naskladnění výrobku a skladník, pracující ve skladu mohl zboží připravit a oznámit, že je zboží připraveno.

Cílem je vytvoření stručného konceptu za pomocí diagramu tříd, use case diagramu, diagramu aktivit a pro doplnění sekvenčním diagramem. Diagramy jsou vytvořeny za pomocí software Microsoft Office, Visio.

DIAGRAM TŘÍD

Diagram tříd modulu skladu:



Obrázek 31: Diagram tříd modulu skladu, zdroj: vlastní zpracování

Diagram pro modul skladu na obrázku 31 obsahuje čtyři základní třídy. Každá třída má vlastní název, který je napsán v modré hlavičce. Ve střední části jsou uvedeny atributy třídy a spodní část obsahuje metody.

TŘÍDY

Třída **pracovník** představuje jak pracovníka, který zadává požadavky, tak i skladníka, který požadavky vyřizuje. Pro jejich odlišení se využívají odlišná práva, která zajišťuje atribut typID. Třída umožňuje pracovníka za pomocí metod zobrazit, upravit smazat a ukládat změny.

Třída **pozadavek** představuje katalog požadavků, kam pracovníci ukládají své žádosti ohledně skladu. Tyto požadavky je možno díky metodám zobrazovat, vytvářet, mazat, upravovat a ukládat.

Třída **vyrobek** představuje výrobky, které jsou uloženy ve skladu a jsou požadovaný pracovníky za pomocí katalogu požadavků. Výrobky lze opět zobrazovat, upravovat, přidávat a mazat, jednotlivé kroky lze ukládat.

Třída **sklad** představuje reálný sklad, v němž jsou uloženy výrobky. Ten lze za pomocí metod upravovat a změny ukládat.

Vztahy

V diagramu tříd jsem využil 3 základní vztahy mezi třídami. Jsou to kompozice, agregace asociace.

Asociace je vztah mezi třídami, který specifikuje spojení mezi jejich instancemi. Tento vztah jsem využil mezi třídami pracovnik a pozadavek.

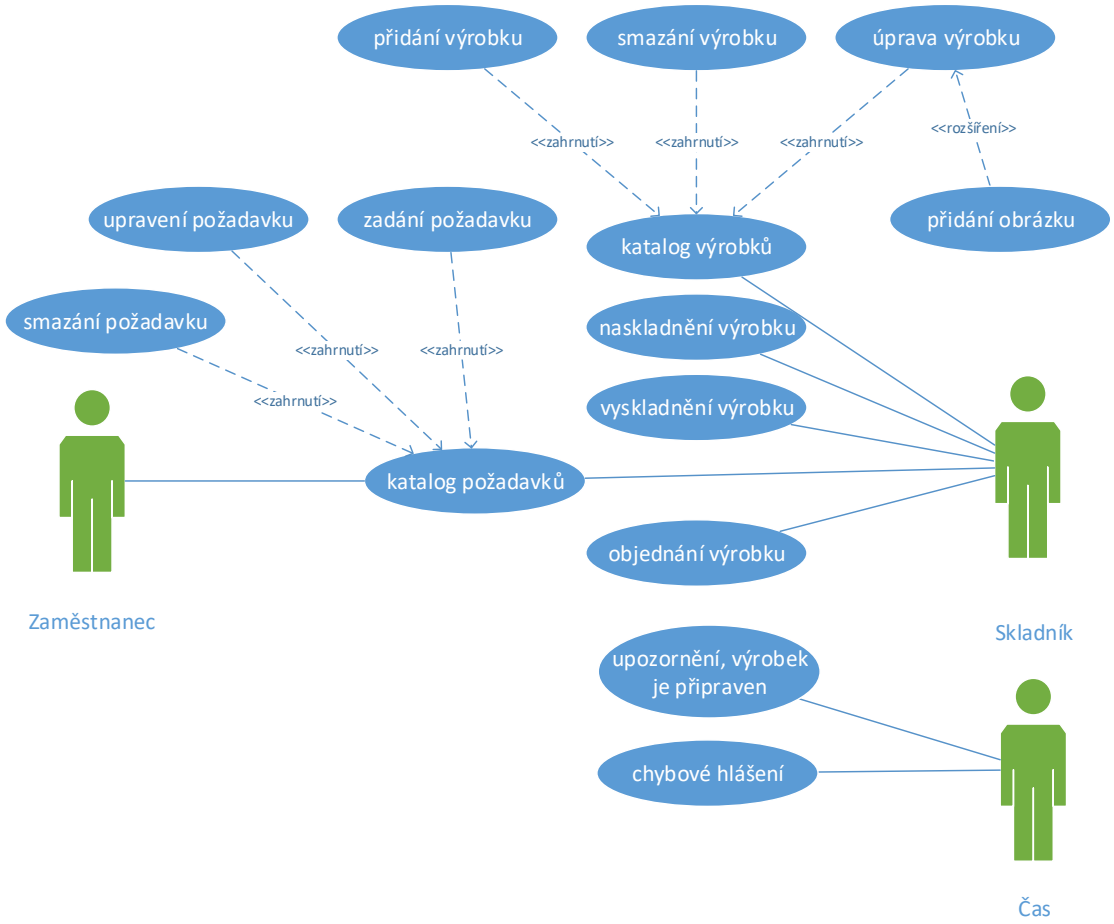
Kompozice je určitá forma asociace, kde je vyjádřen vztah celek část. Tento vztah je využit mezi vyrobkem a pozadavkem.

Agregace je silnější vztah než asociace, *„při zániku kontejneru automaticky rušíme i daný element.“* V diagramu jej nalezneme mezi výrobkem a skladem.

PŘÍPADY UŽITÍ – USE CASE DIAGRAM

„Use case diagram“ zobrazuje informační systém jako celek a ukazuje, v jakých případech jej lze použít. Nejdůležitějšími prvky diagramu jsou hranice systému, aktéři, činnosti a vazby mezi nimi.

Use case diagram modulu skladu (viz obrázek 32):



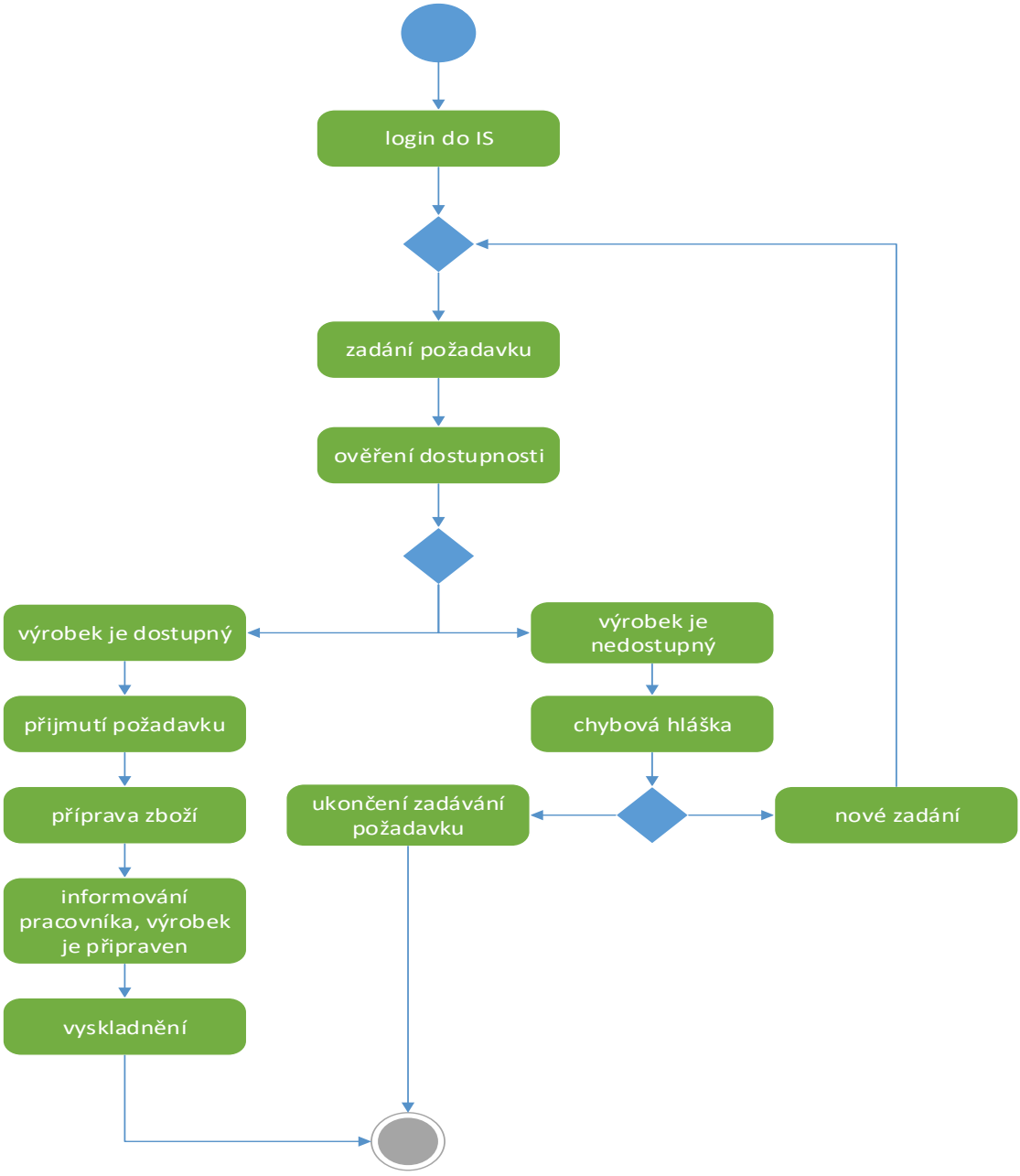
Obrázek 32: USE CASE diagram modulu skladu, zdroj: vlastní zpracování

Aktéři – v modulu sklad se vyskytují tři aktéři. Prvním je zaměstnanec, ten může zadávat požadavky do katalogu požadavků, dále je mazat a upravovat. Do katalogu požadavků má dále přístup i druhý aktér, skladník. Ten dále může výrobky naskladňovat, vyskladňovat, objednávat a má přístup do katalogu výrobků, kde může výrobky měnit, mazat, přidávat. Posledním aktérem je čas, který vykonává činnosti v určitém časovém bodě, např. poskytuje, že je výrobek připraven k vyzvednutí nebo podává chybová hlášení.

V „Use case diagramu“ byly použity relace zahrnutí (include) a rozšíření (extend). *„Relace <<include>> vyčleňuje kroky společníka několika případům užití do samostatného případu užití, který je následně do příslušných užití zahrnut“. „Relace <<extend>> je způsobem, jímž lze do existujícího případu užití vložit nové chování“*

DIAGRAM AKTIVIT

Diagram aktivit popisuje chování systému, konkrétně na obrázku 33 je zachycen proces vyskladnění výrobku.



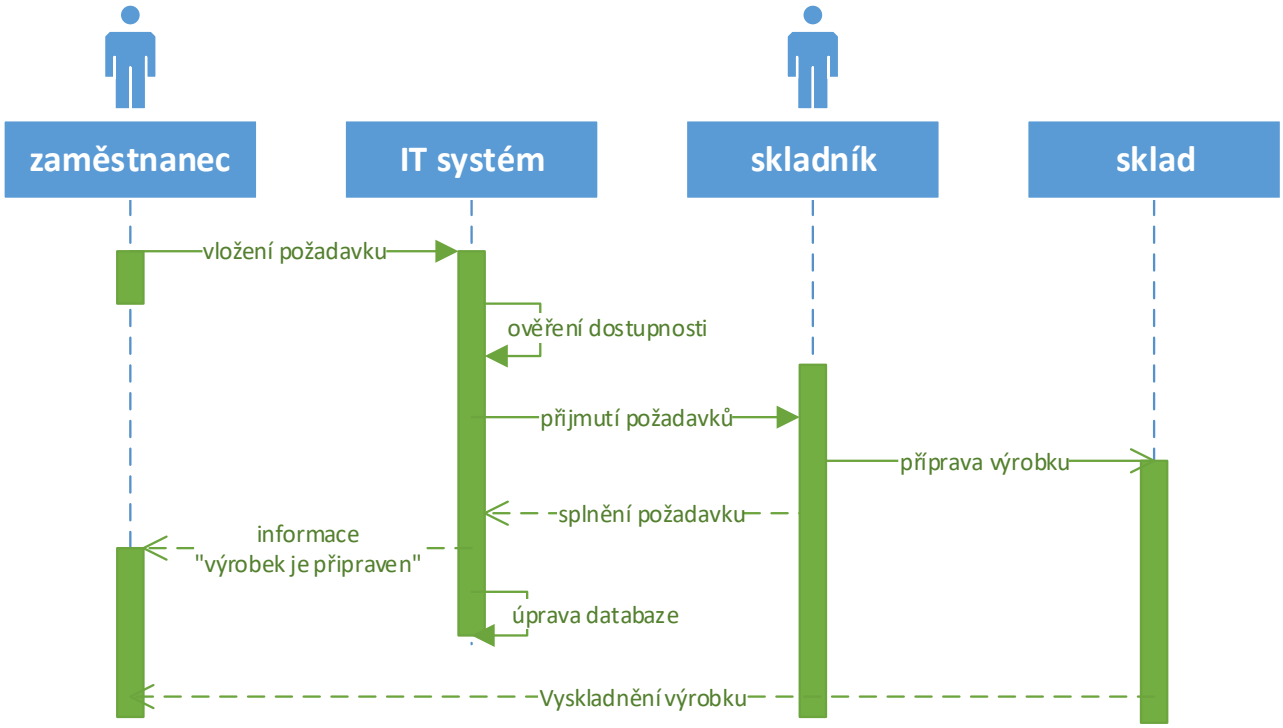
Obrázek 33: Diagram aktivit, zdroj: vlastní zpracování

Diagram se skládá z akcí, označených zelenými políčky, počátečního a koncového uzlu. Tok akcí je zobrazen za pomocí čar se šipkami. Symbol modrého kosočtverce označuje rozhodnutí nebo sloučení.

SEKVENČNÍ DIAGRAM

Sekvenční diagram ukazuje posloupnost daných akcí, jak následují postupně v čase. Diagram je navržen tak, aby byl dodržen základní logický koncept. Vynechání jedná části má ve většině případů zhroucení celého systému.

Diagram na obrázku 34 ukazuje, jak vypadá úspěšné vyskladnění výrobku ze skladu.



Obrázek 34: Sekvenční diagram, zdroj: vlastní zpracování

Modré pole s panáčkem vyjadřují životnost aktéra, modrá políčka bez panáčka vyjadřují životnost objektů. Zelená svislá pole ukazují, kdy je objekt aktivní. Dané aktivity (sekvence) jsou pak značeny čárami se šipkami, které udávají směr. Plná čára označuje klasickou zprávu, přerušovaná čára pak zprávu návratovou a šipka zalomená zpět označuje zpětnou zprávu (např. IT systém sám zjistí, jestli je výrobek dostupný a sám upraví data v databázi).

10.2 Podnikový prodej - vytvoření nové objednávky zákazníkem

SLOVNÍ POPIS PODNIKOVÉHO PROCESU

Jedním z nejdůležitějších podnikových procesů je Prodej. V našem případě se firma zabývá nákupem a následným prodejem zboží skrze E-commerce systém. Celý tento podnikový proces začíná tím, že vedení firmy zadá do informačního systému pokyn k zahájení marketingové kampaně. Informační systém informuje manažera prodeje o pokynu vedení firmy (např. současně skrz notifikační lištu v systému, sms, emailem, atd.). Manažer následně zadává do informačního systému pokyn, který je určený pro oddělení prodeje. Systém informuje oddělení prodeje o pověření k zahájení marketingové kampaně. Oddělení tedy zahájí marketingovou kampaň a kontaktuje potencionální zákazníky s nabídkou produktů (může se jednat o emailový kontakt, telefonický kontakt či jinou formu). Pokud má zákazník o zboží zájem, oddělení prodeje vloží do systému žádost o odeslání odkazu na katalog produktů, resp. webové rozhraní E-commerce systému. Systém následně odešle zákazníkovi zprávu obsahující odkaz na E-commerce systém. Později se zákazník naviguje na webové rozhraní E-commerce. V tom momentě E-commerce navyšuje počítadla návštěvnosti a informuje systém. Systém si to poznamená a v požadovaných intervalech (měsíčně, týdně, atd.) generuje sestavy návštěvnosti, které jsou určené vedení firmy. E-commerce po připojení zákazníka předkládá zákazníkovi žádost o registraci. Zákazník se registruje a E-commerce registraci potvrzuje. Poté je zákazníkovi předložen katalog zboží. Při výběru zboží zákazníka kontroluje E-commerce skrz dotaz na sklad, zda je zboží dostupné. E-commerce poté odesílá systému výběr zákazníka, který je určen k analýze potenciálně chtěného zboží, na které se později mohou vztahovat slevy. Systém opět generuje sestavy přehledu potencionálně chtěných výrobků a ty předkládá vedení firmy. Současně informuje manažera prodeje o potencionálně velké objednávce. Manažer prodeje konzultuje s vedením firmy případné slevy. Vedení firmy dává pokyn manažerovi k zadání hromadné slevy pro objednávku zákazníka. Manažer vkládá hromadnou slevu do E-commerce systému ke konkrétní objednávce. E-commerce generuje konečnou cenu, způsob dopravy a platby. Zákazník vybírá údaje. E-commerce přesměrovává zákazníka na platební bránu. Zákazník provádí platbu. Platební brána informuje jak prodejce, tak zákazníka o provedené platbě. E-commerce generuje fakturu zákazníkovi a informuje systém o nové objednávce. Systém informuje oddělení prodeje o nové objednávce ke zpracování.

Pro přehlednost je uvedený proces přepsán do tabulky 4 scénáře případu užití:

Pořadí činnosti	**Kdo** (počáteční aktér)	**Akce**	**Cíl** (cílový aktér)
1.	**Vedení firmy**	*Zadá do informačního systému pokyn k zahájení marketingové kampaně*	**Systém**
2.	**Systém**	*Informuje manažera prodeje o pokynu vedení firmy*	**Manažer prodeje**
3.	**Manažer prodeje**	*Zadává do informačního systému pokyn určený pro oddělení prodeje*	**Systém**
4.	**Systém**	*Informuje oddělení prodeje o pověření k zahájení marketingové kampaně*	**Oddělení prodeje**
5.	**Oddělení prodeje**	*Kontaktuje potenciálního zákazníka ohledně katalogu produktů*	**Zákazník**
6.	**Zákazník**	*Projevuje zájem o zboží*	**Oddělení prodeje**
7.	**Oddělení prodeje**	*Vkládá do systému žádost o odeslání odkazu na katalog produktů*	**Systém**
8.	**Systém**	*Odesílá odkaz na katalog produktů*	**Zákazník**
9.	**Zákazník**	*Navigace na webové rozhraní*	**E-commerce**
10.	**E-commerce**	*Navyšuje počítadla návštěvnosti*	**Systém**
11.	**Systém**	*Generuje sestavu návštěvnosti webu*	**Vedení firmy**
12.	**E-commerce**	*Předkládá zákazníkovi žádost o registraci*	**Zákazník**
13.	**Zákazník**	*Provádí registraci*	**E-commerce**
14.	**E-commerce**	*Potvrzuje registraci*	**Zákazník**
15.	**E-commerce**	*Předkládá nabídku zboží*	**Zákazník**
16.	**Zákazník**	*Vybírá zboží*	**E-commerce**
17.	**E-commerce**	*Kontroluje dostupnost zboží na skladě*	**Sklad**
18.	**Sklad**	*Potvrzuje dostupnost zboží*	**E-commerce**
19.	**E-commerce**	*Odesílá výběr zákazníka k analýze*	**Systém**
20.	**Systém**	*Generuje sestavu potencionálně chtěného zboží*	**Vedení firmy**
21.	**Systém**	*Informuje manažera o potencionálně velké objednávce*	**Manažer prodeje**
22.	**Manažer prodeje**	*Konzultuje s vedením potencionální slevy objednávky*	**Vedení firmy**
23.	**Vedení firmy**	*Zadává pokyn k úpravě smluvních cen v objednávce*	**Manažer prodeje**
24.	**Manažer prodeje**	*Zadává hromadnou slevu pro objednávku*	**E-commerce**
25.	**E-commerce**	*Generuje konečné ceny, způsob dopravy a platby*	**Zákazník**
26.	**Zákazník**	*Vybírá možnosti*	**E-commerce**
27.	**E-commerce**	*Přesměrovává zákazníka na platební bránu*	**Zákazník**
28.	**Zákazník**	*Uskutečňuje platbu*	**Platební brána**
29.	**Platební brána**	*Informuje zákazníka o uskutečněné platbě*	**Zákazník**
30.	**Platební brána**	*Informuje E-Commerce o platbě*	**E-Commerce**
31.	**E-commerce**	*Generuje Fakturu*	**Zákazník**
32.	**E-commerce**	*Informuje systém o nové objednávce*	**Systém**
33.	**Systém**	*Informuje oddělení prodeje o objednávce ke zpracování*	**Oddělení prodeje**

Tabulka 4: Scénář případu užití, zdroj: vlastní zpracování

USE CASE DIAGRAMY

Celkově tedy máme v modulu vytvoření objednávky tyto aktéry, kteří vykonávají následující operace (viz Tabulka 5):

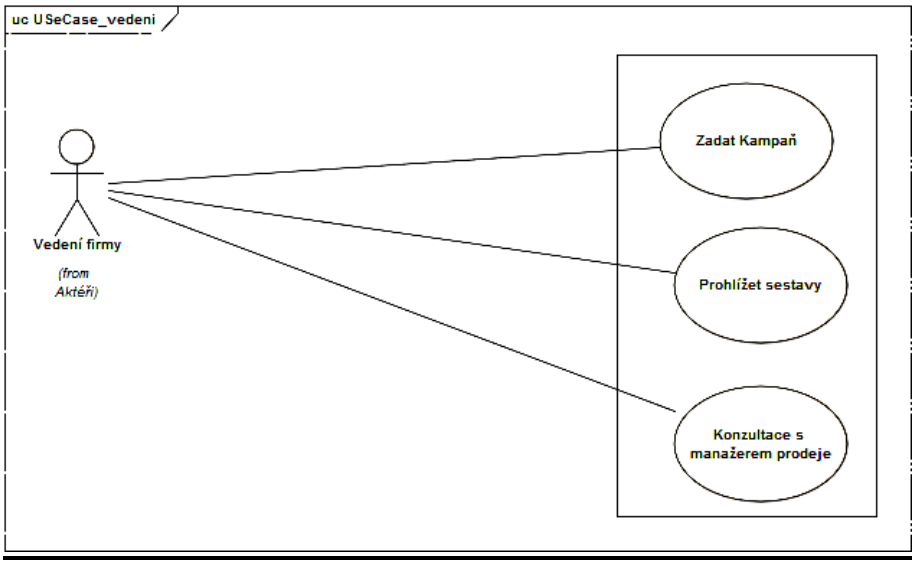
Aktér	Akce
Vedení firmy	*Zadat kampaň* *Prohlížet sestavy* *Konzultace s manažerem prodeje*
Manažer prodeje	*Zahájit kampaň* *Interakce s E-commerce* *Konzultace s vedením firmy*
Oddělení prodeje	*Spustit kampaň* *Zadat požadavek na odeslání produktu* *Řešit nové objednávky*
Systém	*Informovat o kampani* *Analyzovat výběr zákazníka* *Generovat sestavy*
E-commerce	*Interakce se zákazníkem* *Interakce se systémem* *Interakce s platební bránou* *Interakce s manažerem prodeje* *Interakce se skladem*
Zákazník	*Interakce s oddělením prodeje* *Interakce s E-commerce* *Interakce s platební bránou*
Platební brána	*Interakce se zákazníkem* *Interakce s E-commerce*
Sklad	*Interakce s E-commerce*

Tabulka 5: Aktéři a akce modulu vytvoření objednávky, zdroj: vlastní zpracování

PŘEHLED USE CASE DIAGRAMŮ:

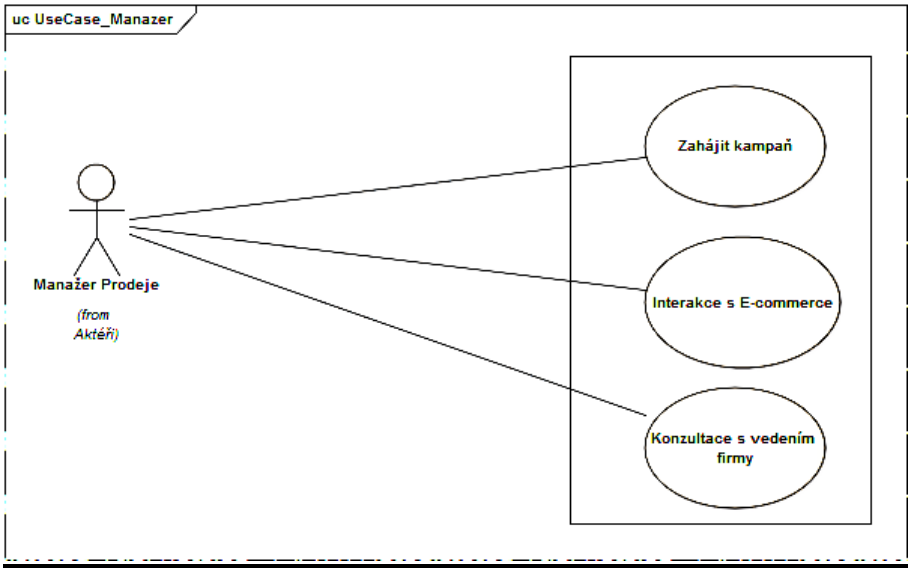
Na obrázcích 35 až 42 jsou nakresleny Use Case diagramy pro jednotlivé aktéry informačního systému „Podnikový prodej – vytvoření objednávky“.

Vedení firmy:



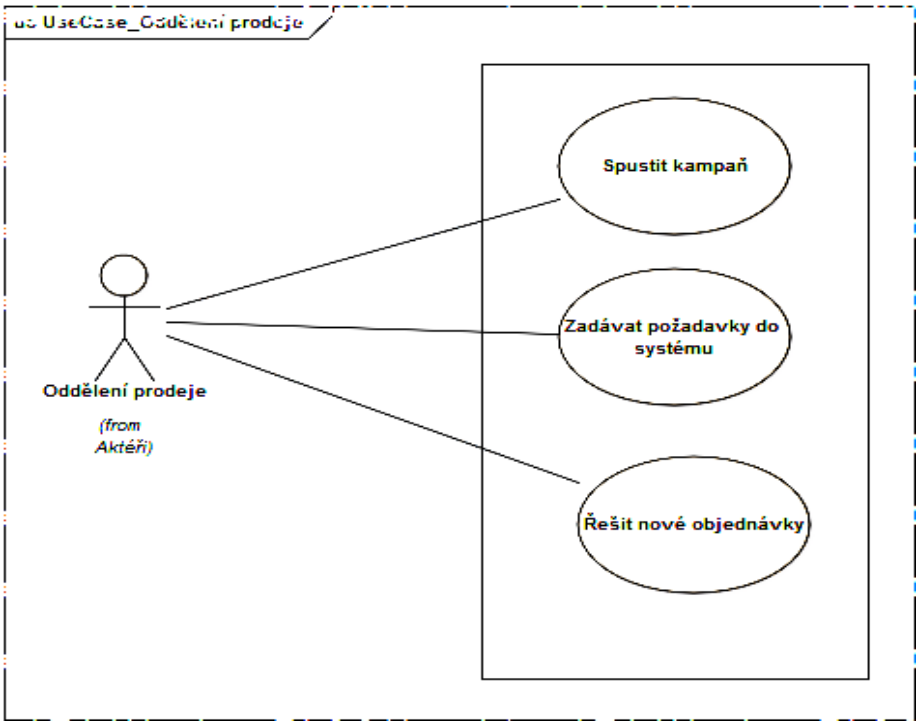
Obrázek 35: USE CASE diagram – vedení firmy, zdroj: vlastní zpracování

Manažer prodeje:



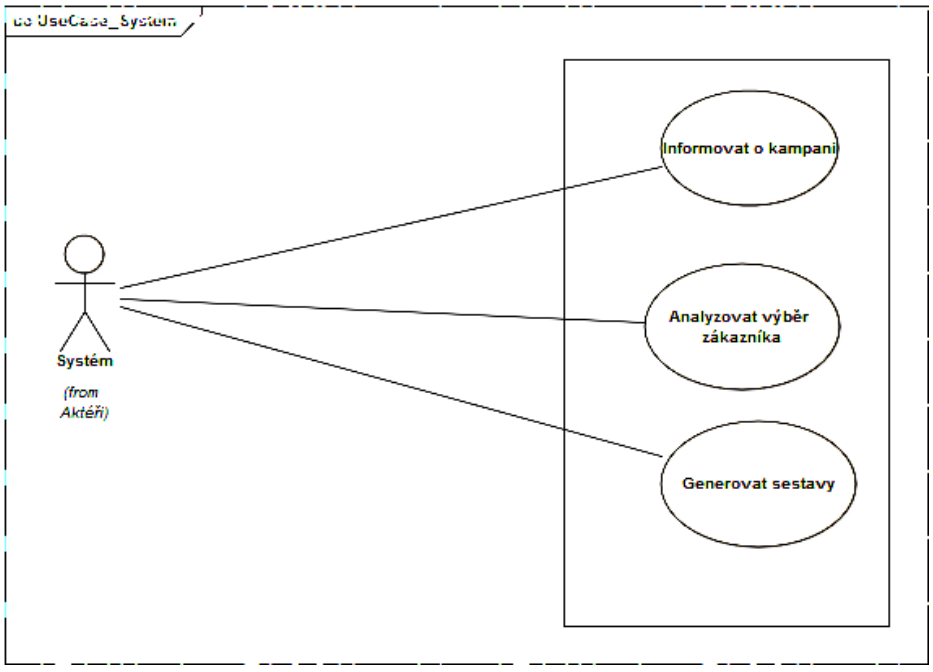
Obrázek 36: USE CASE diagram – manažer prodeje, zdroj: vlastní zpracování

Oddělení prodeje:



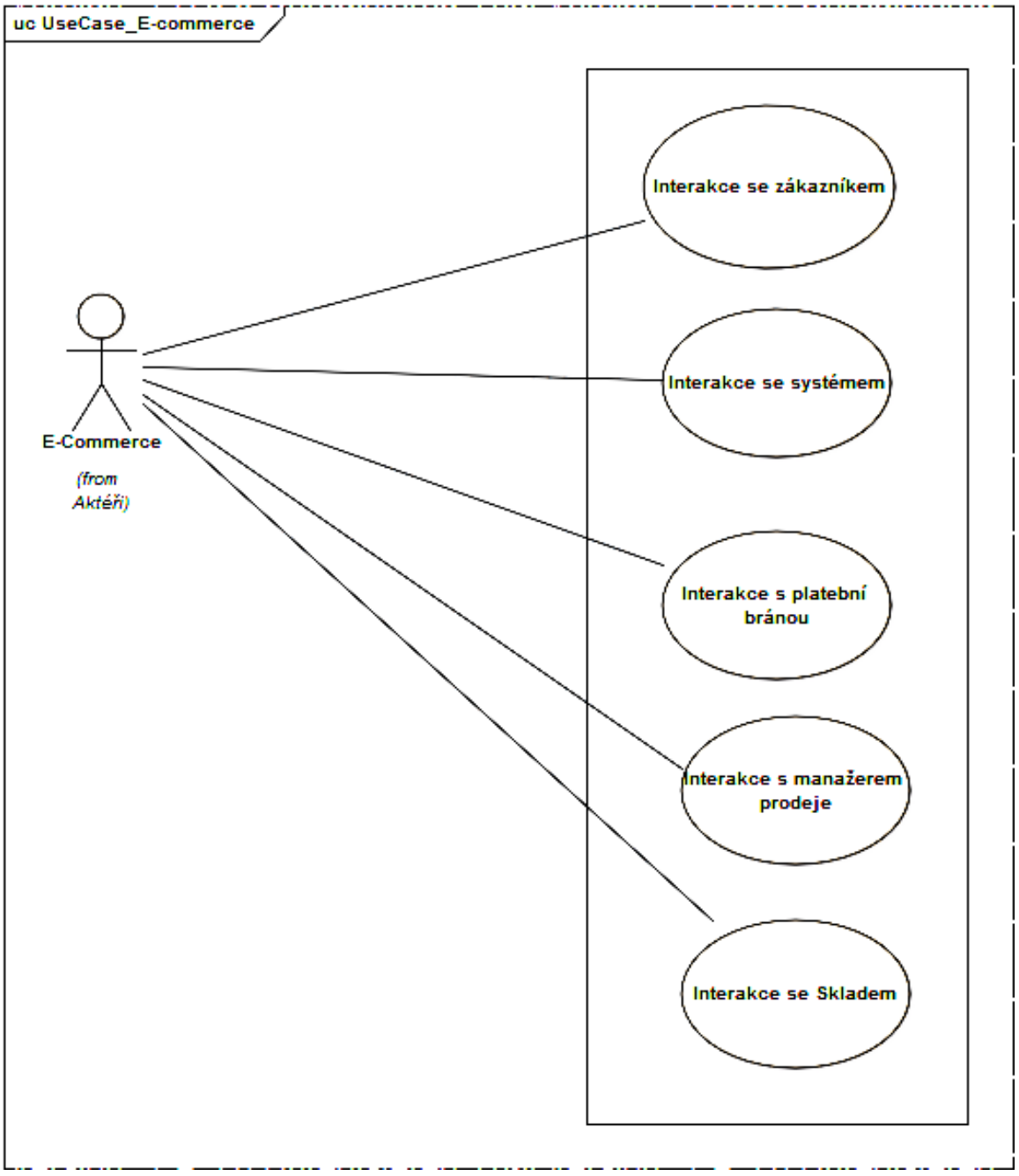
Obrázek 37: USE CASE diagram – prodejní oddělení, zdroj: vlastní zpracování

Systém:



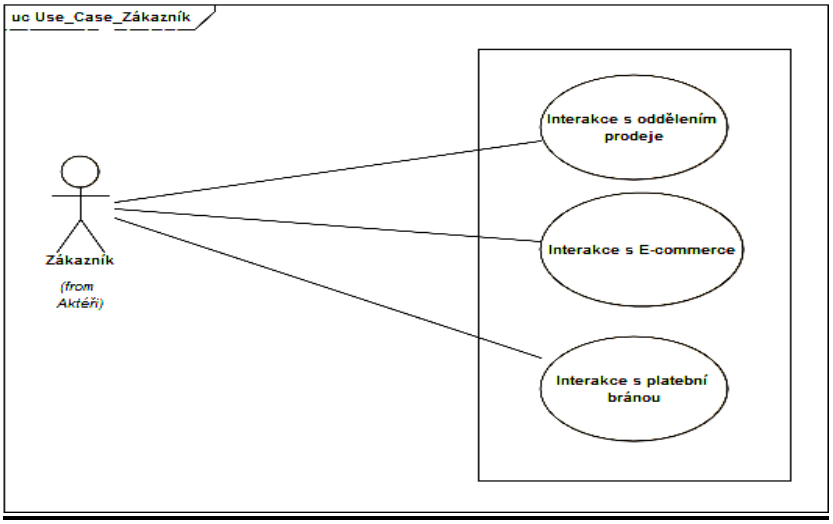
Obrázek 38: USE CASE diagram – systém, zdroj: vlastní zpracování

E-commerce



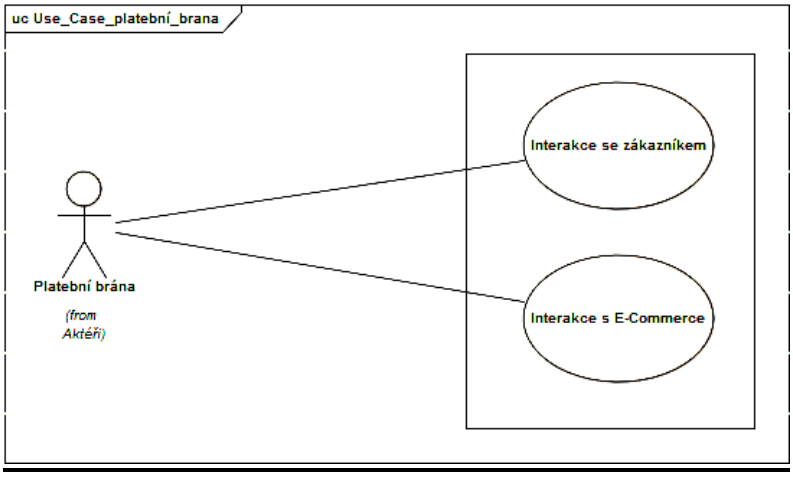
Obrázek 39: USE CASE diagram pro E-commerce, zdroj: vlastní zpracování

Zákazník:



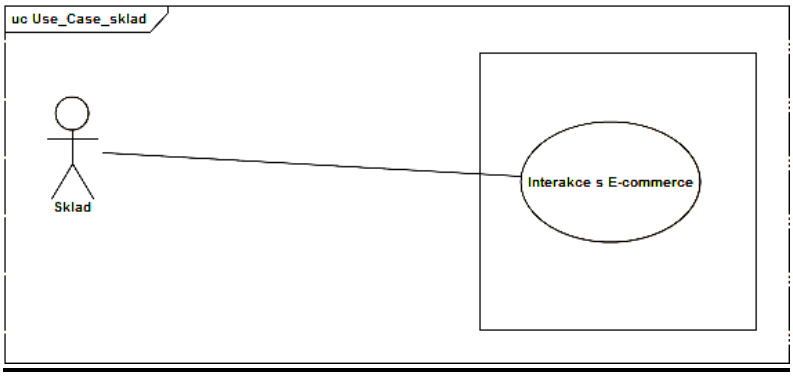
Obrázek 40: USE CASE diagram - Zákazník, zdroj: vlastní zpracování

Platební brána:



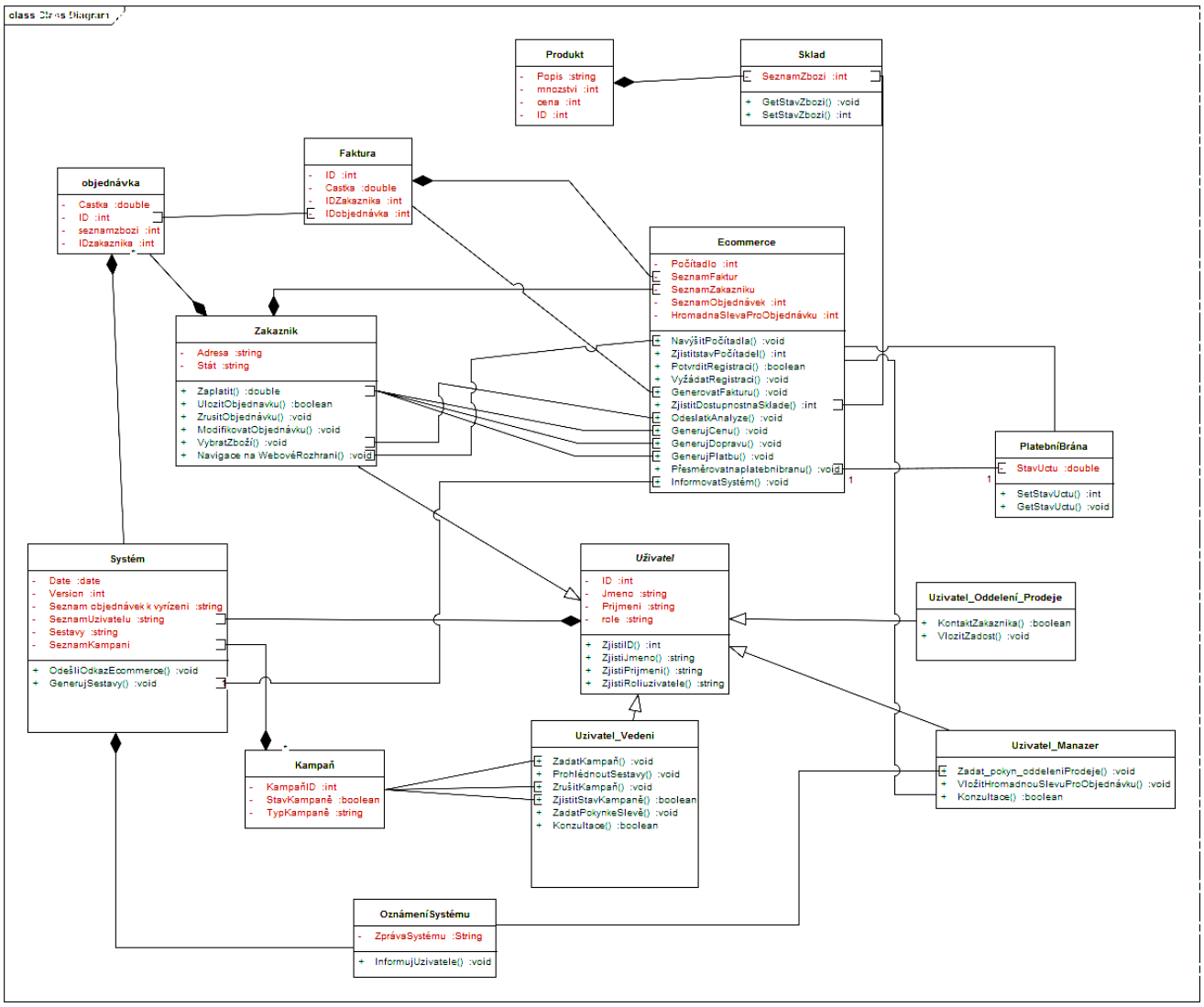
Obrázek 41: USE CASE diagram – Platební brána, zdroj: vlastní zpracování

Sklad:



Obrázek 42: USE CASE diagram - Sklad, zdroj: vlastní zpracování

DIAGRAM TŘÍD

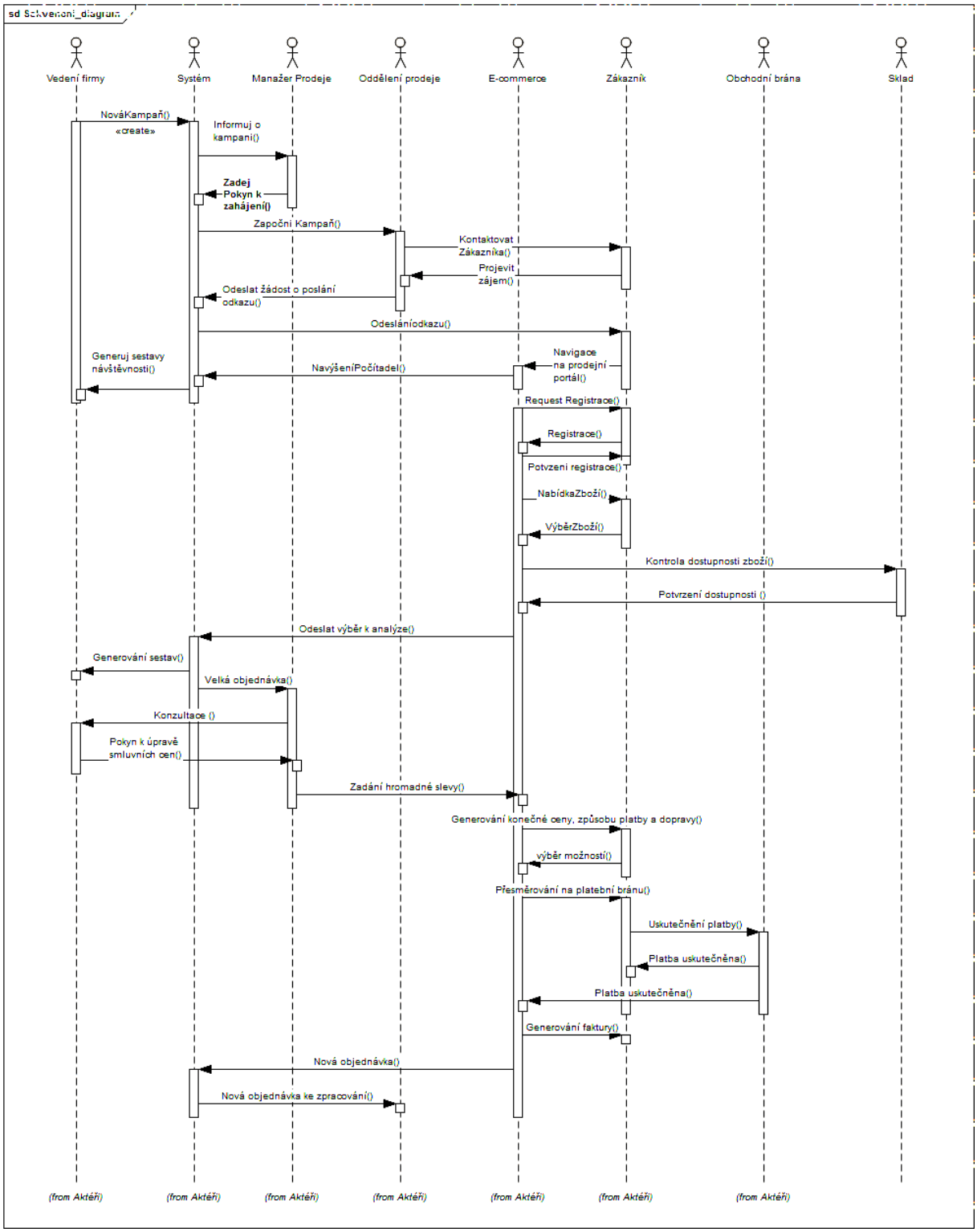


Obrázek 43: Diagram tříd, zdroj: vlastní zpracování

Výše znázorněný diagram tříd na obrázku 43 zachycuje vztahy mezi třídami, konkrétně asociaci, agregaci a kompozici. Jedná se o jakýsi předběžný model, neboť ve finální verzi musí být mnohem propracovanější.

SEKVENČNÍ DIAGRAM

Sekvenční diagram pokrývá za sebou jdoucí činnosti v podniku. V tomto diagramu na obrázku 44 mají jednotliví aktéři přiděleny plovoucí dráhy tzv. swimlines. Mezi těmito aktéry probíhají činnosti formou zpráv. Na rozdíl od předchozích diagramů, jedná se o diagram dynamický.



Obrázek 44: Sekvenční diagram, zdroj: vlastní zpracování

10.3 Modelování IS knihovny

Úvod

Studie je zaměřena na problematiku informačního systému knihovny. Jedná se o jednoduchý informační systém, který by měl nabízet základní funkce, jak pro registrované a neregistrované uživatele a administrátora. Modelování tohoto zjednodušeného informačního systému řeší funkce a vztahy mezi registrovanými uživateli, knihami a jejich rezervacemi a výpůjčkami. Důležitou roli při návrhu informačních systému je přijat na veškeré komponenty a části, které budou následně uživatelé a administrátoři využívat a přijít na způsob, které z těchto složek a jakým způsobem provázat, aby spolu bezchybně komunikovaly a také nezatěžovaly zbytečně systém

Problém nastává při půjčování knih, kdy se některé tituly nevracejí, a po delší době je složité dohledat, kdo si daný titul vypůjčil.

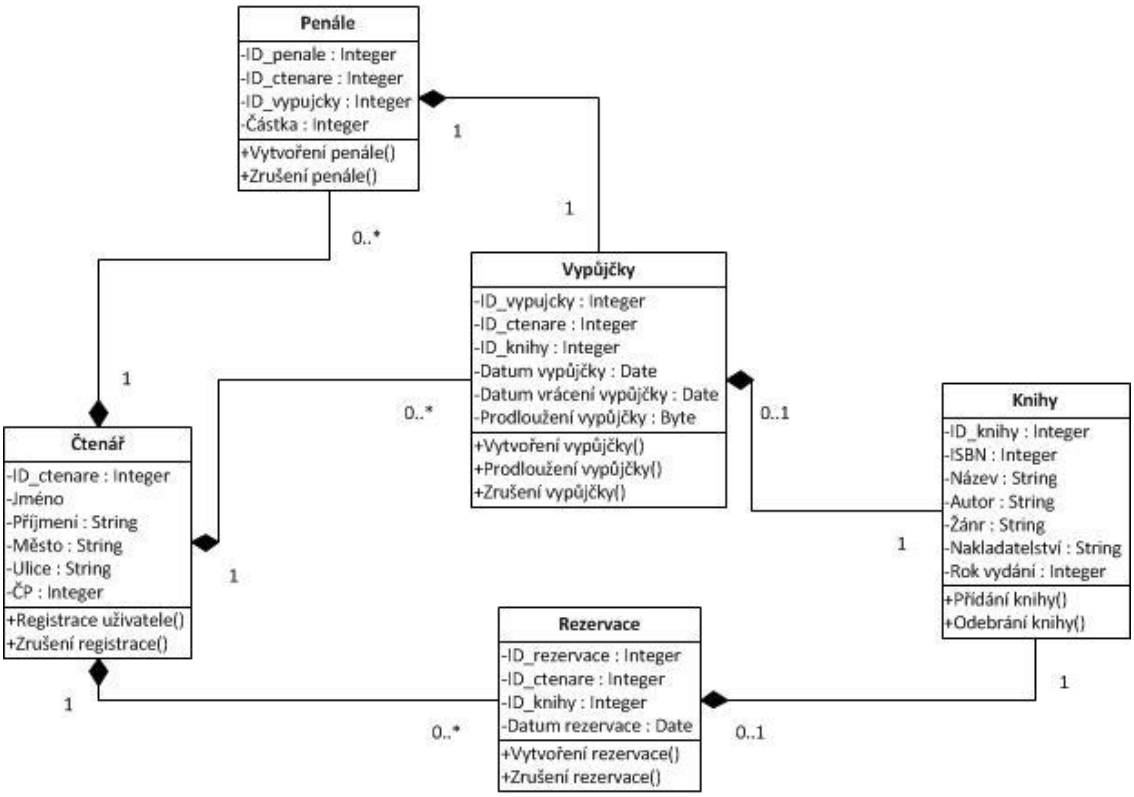
Při modelování bylo využito produkt Microsoft Visio.

Funkce IS:

Prohlížení titulů, Registrace uživatele, Přihlašování uživatelů, Rezervace titulů, Správa registrovaných uživatelů, Správa titulů, Správa vypůjčených titulů

Diagram tříd

Diagram na obrázku 45 znázorňuje datový model IS, jeho třídy a vztahy mezi nimi. U jednotlivých tříd jsou poté popsány jejich atributy a operace.



Obrázek 45: Diagram tříd IS, zdroj: vlastní zpracování

USE CASE DIAGRAM

Diagram případu užití na obrázku 46 zobrazuje procesy spojené se správou knih, rezervací, výpůjček a čtenářů. V diagramu jsou zobrazeny tři základní role host, čtenář a administrátor.

Host

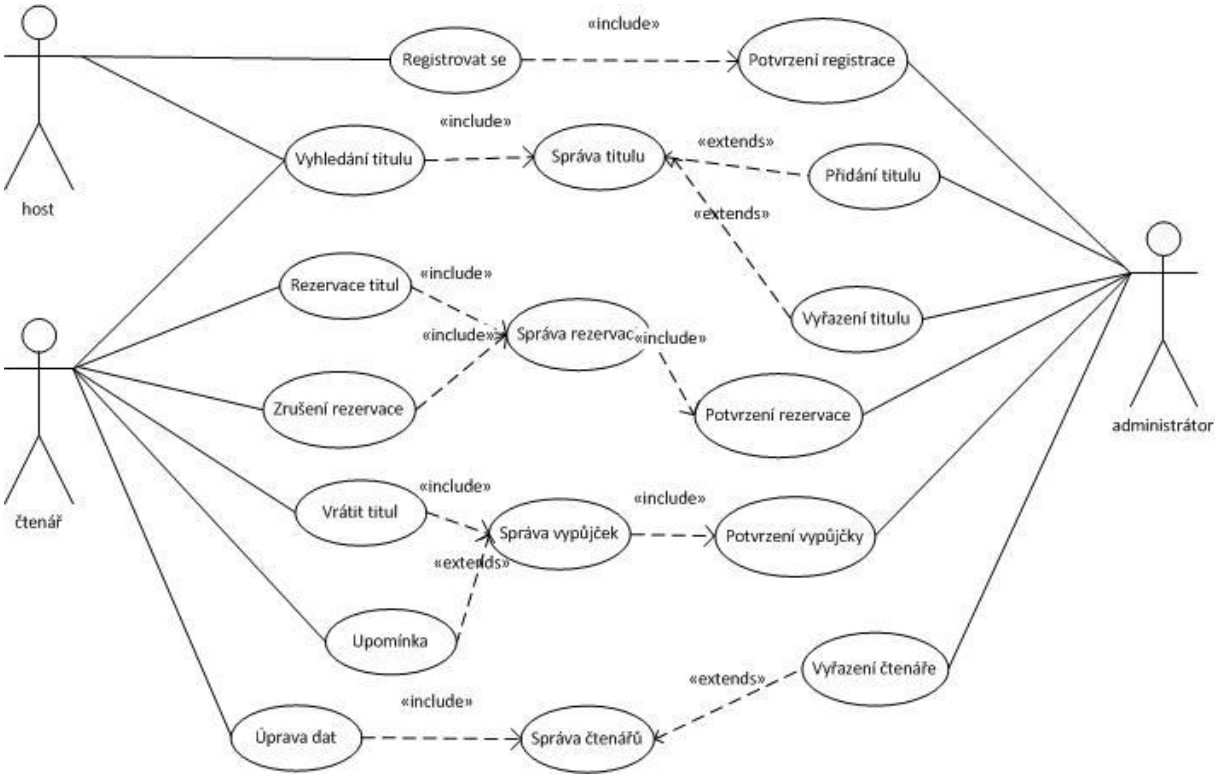
Jedná se o běžného uživatele, který není přihlášen. Tudíž má omezené možnosti, co se týče využívání systému. Jedinými funkcemi, které jsou mu k dispozici je vyhledávání knih v databázi a možnost se zaregistrovat do systému.

Čtenář

Tato role náleží již přihlášenému uživateli. Tento již může, kromě vyhledávání knih, také provádět rezervace titulů, rušit rezervace, spravovat své výpůjčky (vrácení knih, placení za upomínky) a spravovat informace na svém čtenářském profilu.

Administrátor

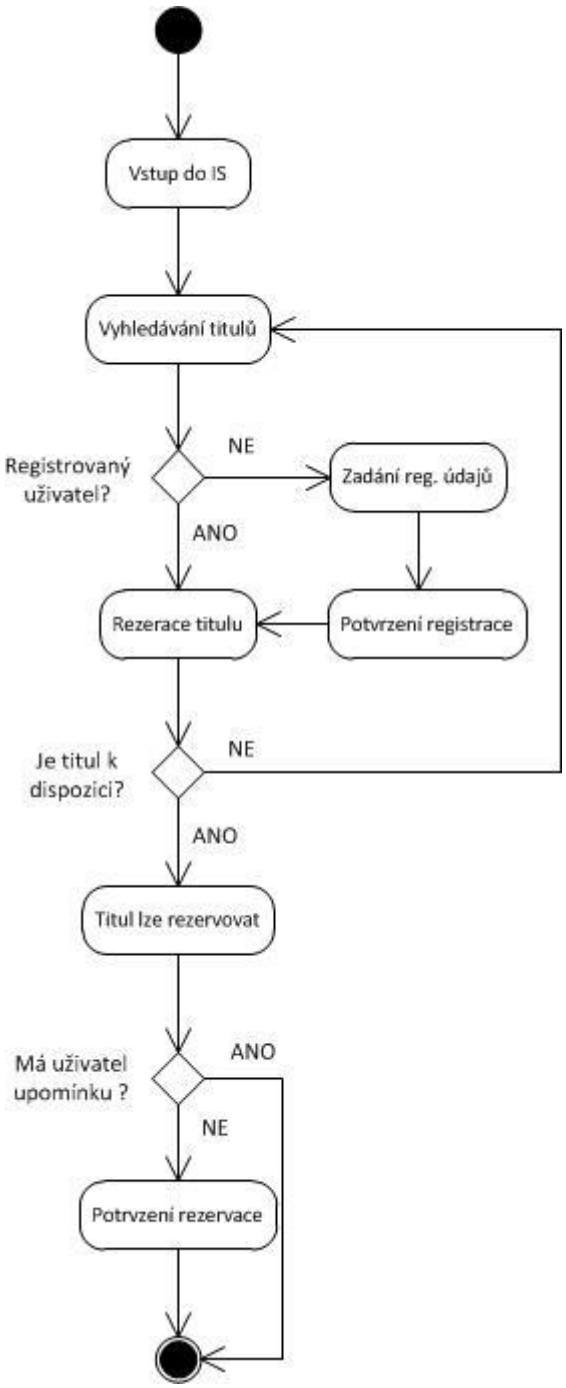
Administrativní pracovník, může v systému využívat služeb přidávat a vyřazovat tituly či čtenáře, potvrzovat rezervace, výpůjčky a registrace uživatelů.



Obrázek 46: USE CASE diagram – IS knihovny, zdroj: vlastní zpracování

DIAGRAM AKTIVITY

Diagram na obrázku 47 zobrazuje užití informačního systému, kdy do něj vstoupí nepřihlášený uživatel a chce si vypůjčit knihu. Systém ho následně vyzve k přihlášení, příp. registraci, a zjistí, zda je titul k dispozici a zda již čtenář nemá nějakou upomínku.



Obrázek 47: Diagram aktivity IS knihovny, zdroj: vlastní zpracování

SCÉNÁŘ PŘÍPADU UŽITÍ

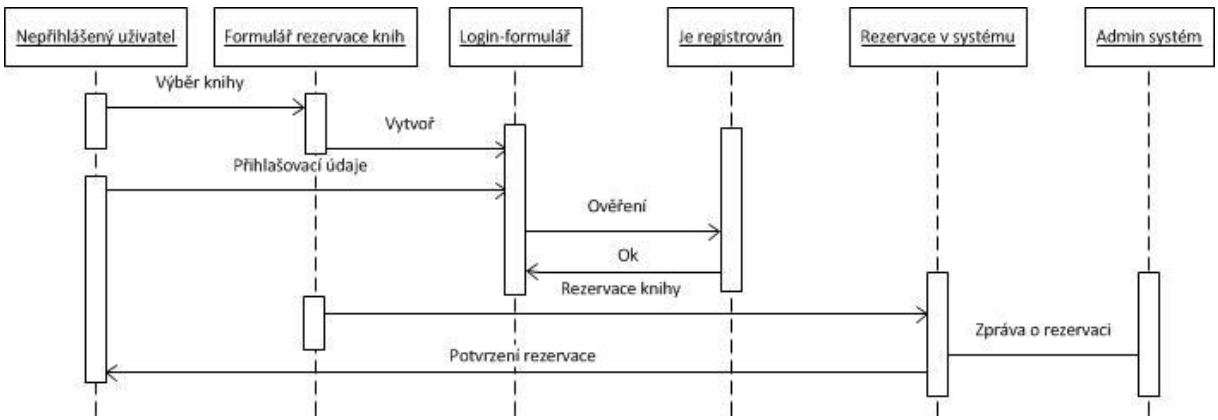
Tato část již slovně popisuje události (tabulka 6), při nichž dochází od vstupu nepřihlášeného uživatele po uložení veškerých dat do databáze systému.

Krok	Role	Scénář
1	Uživatel	Vstoupí do IS
2	Systém	Zobrazí možnost vyhledávání, přihlášení, nebo registrace
3	Uživatel	Zvolí registraci
4	Systém	Zobrazí formulář pro registraci
5	Uživatel	Zadá své údaje a provede registraci
6	Systém	Potvrdí registrací a zobrazí potvrzení
7	Systém	Zobrazí formulář pro přihlášení
8	Uživatel	Zadá své údaje a přihlásí se
9	Systém	Zobrazí uživateli možnost vyhledávání
10	Uživatel	Vybere titul k rezervaci
11	Systém	Provede kontrolu, zda je titul k dispozici a zda nemá uživatel upomínku
12	Systém	Nahlásí uživateli, že titul je možné si rezervovat
13	Uživatel	Potvrdí rezervaci
14	Systém	Uloží údaje o rezervaci do databáze a pošle zprávu administrátorovi

Tabulka 6: Scénář případu užití po vstupu nepřihlášeného uživatele do IS knihovny, zdroj: vlastní zpracování

SEKVENČNÍ DIAGRAM

Na obrázku 48 diagramu je znázorněno, jak bude probíhat komunikace mezi aktérem a komponenty systému v čase.



Obrázek 48: Sekvenční diagram komunikace mezi aktérem a komponenty systému v čase, zdroj: vlastní zpracování

10.4 IS Cestovní kanceláře

Úkolem je realizovat rezervační systém fiktivní cestovní kanceláře (CK).

Tento učebnicový příklad návrhu informačního systému je s malými úpravami a zestručněním převzat z literatury [BUCH2013]. V této vysokoškolské učebnici jsou uvedeny další užitečné postupy modelování a využití metodiky při tvorbě IS.

ZÁKLADNÍ CHARAKTERISTIKA POTŘEB

Základní princip fungování cestovní fiktivní kanceláře SunnyTour je takový, že CK nakoupí od zahraničních partnerů jednotky ubytování a prodává je s příslušným ziskem svým zákazníkům. CK také nabízí dopravu do cílových destinací s pomocí různých dopravců. Sled kroků je následující:

1. V dostatečném předstihu před začátkem sezóny si CK objedná jednotky ubytování v jednotlivých objektech.

2. Ze smluvených cen vytvoří CK pomocí různých kalkulačních nástrojů nabídku zájezdů.

3. Svou nabídku zájezdů a pobytů CK prezentuje veřejnosti.

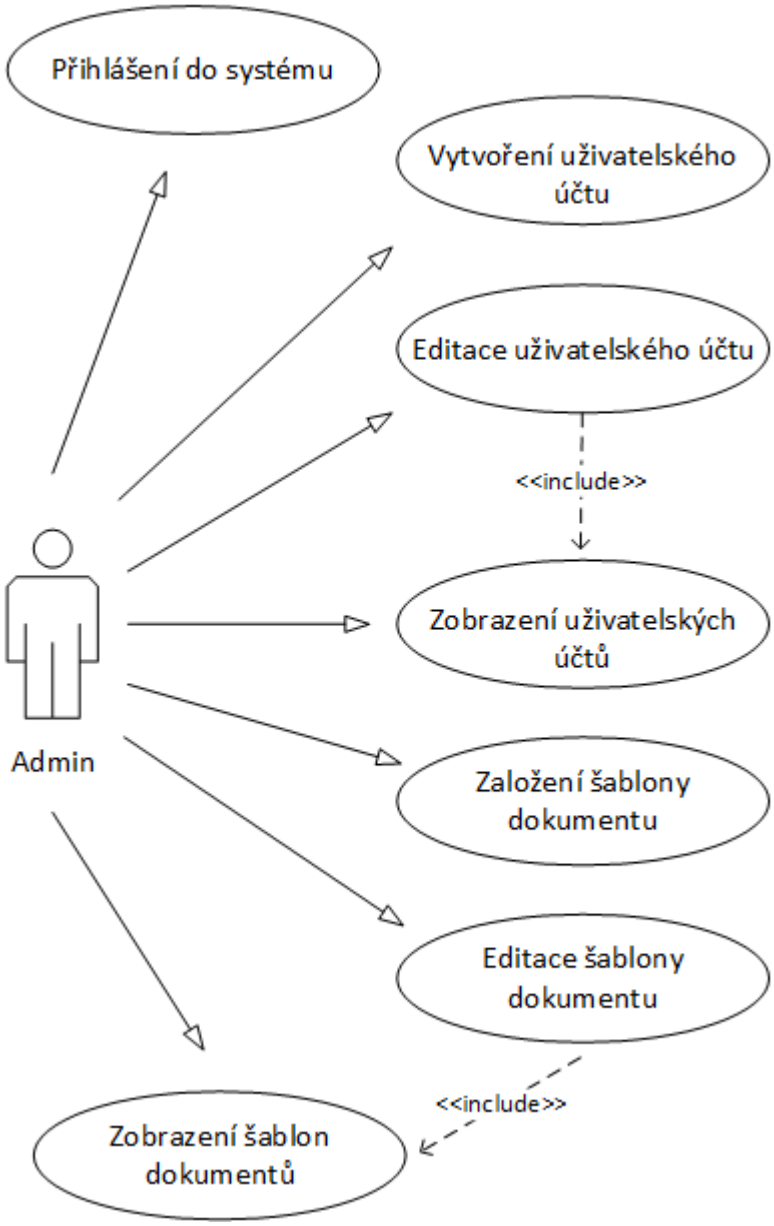
4. Pomocí vhodných nástrojů CK zpracovává objednávky a spravuje obsazenost pokojů a apartmánů v jednotlivých objektech.

5. CK svým zákazníkům vystavuje různé dokumenty, především cestovní smlouvu a vouchery.

6. Do hotelů a apartmánů CK zasílá před každým zájezdem seznam cestujících.

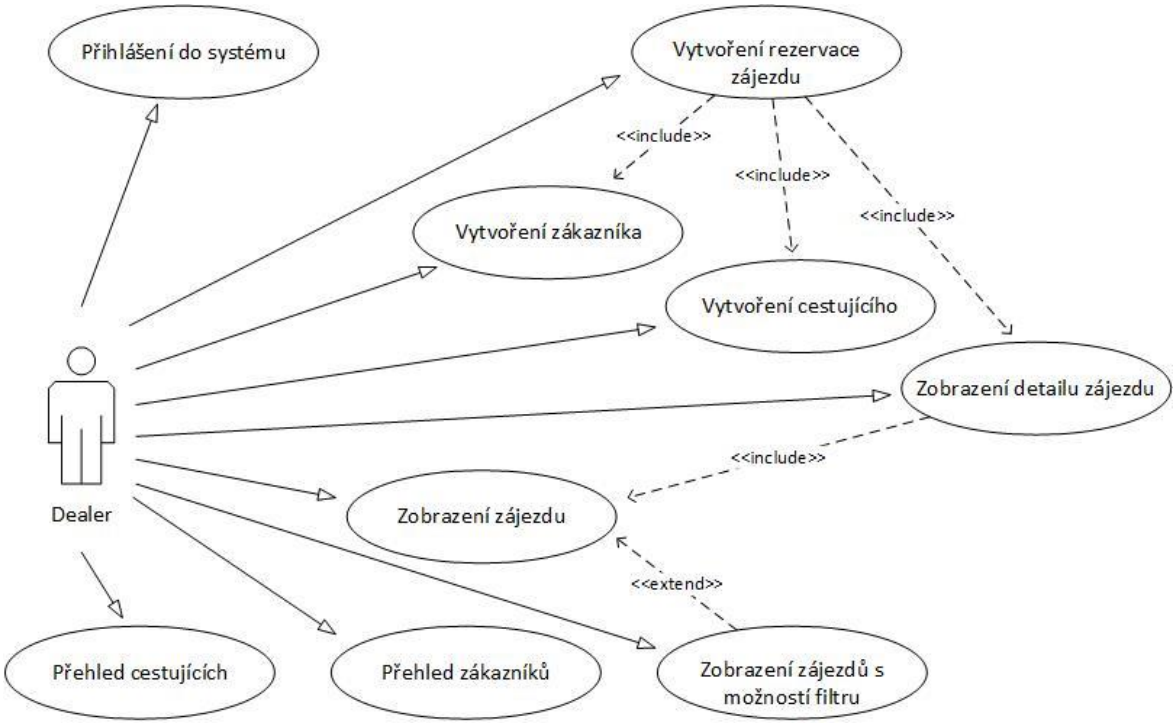
MODELY PŘÍPADU UŽITÍ

Na následujících obrázcích jsou pro jednotlivé aktéry zpracovány konkrétní případy užití.



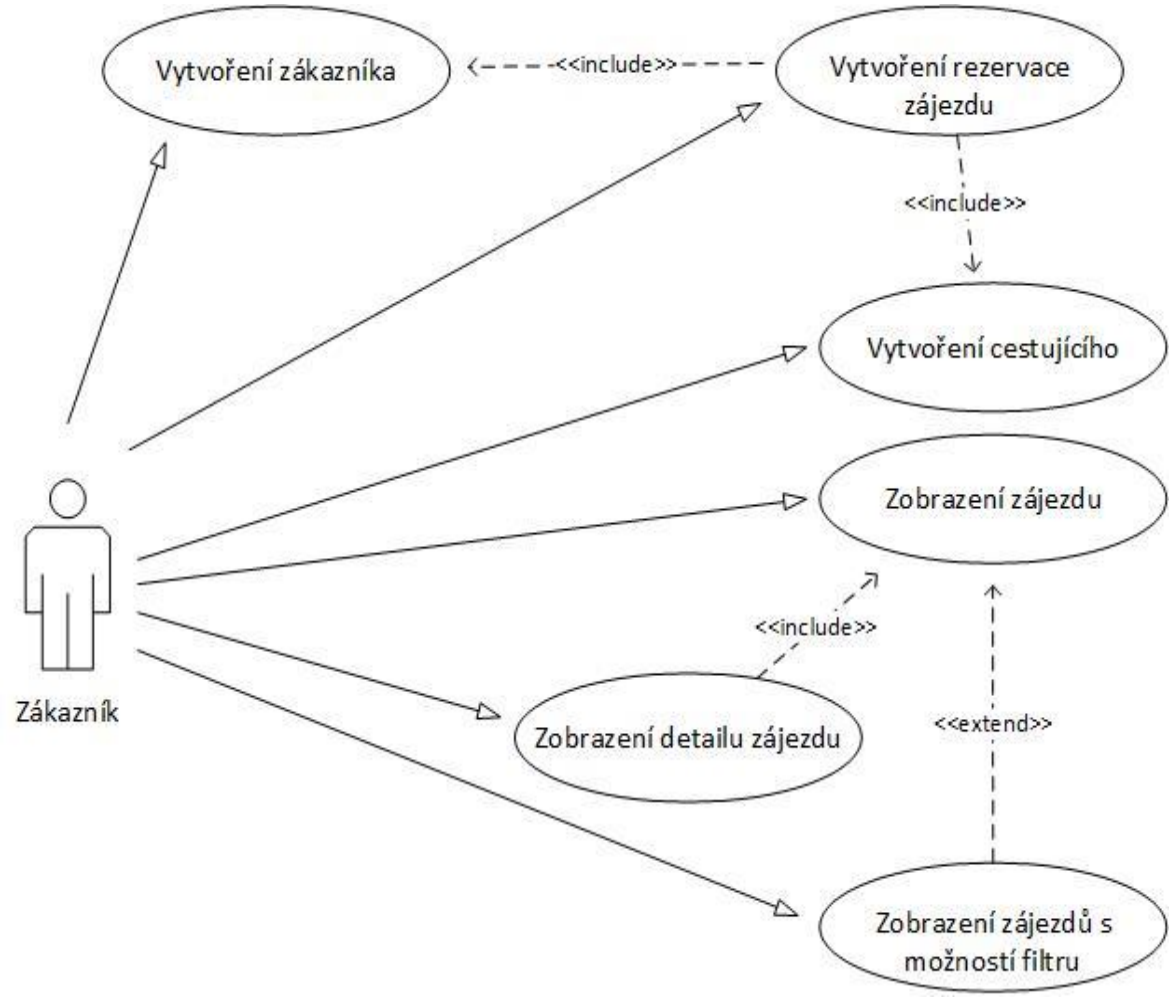
Obrázek 49: USE Case pro aktéra Admin, zdroj: [BUCH2013] a vlastní úprava

Na tomto USE CASE diagramu UML na obrázku 49 jsou zachyceny případy užití aktéra ADMIN. Všechny případy užití spouští aktér, proto je použita asociace se šipkou. Je využit vtah případů užití include, který umožňuje zahrnout, resp. vložit jiný případ užití, jak je popsáno v kapitole USE CASE. Použije se například pro vyjmutí opakující se činnosti do samostatného případu užití“. Případ užití Editace uživatelského konta zahrnuje jako svou nedílnou součást Zobrazení uživatelských kont, proto je zde použit vztah include.



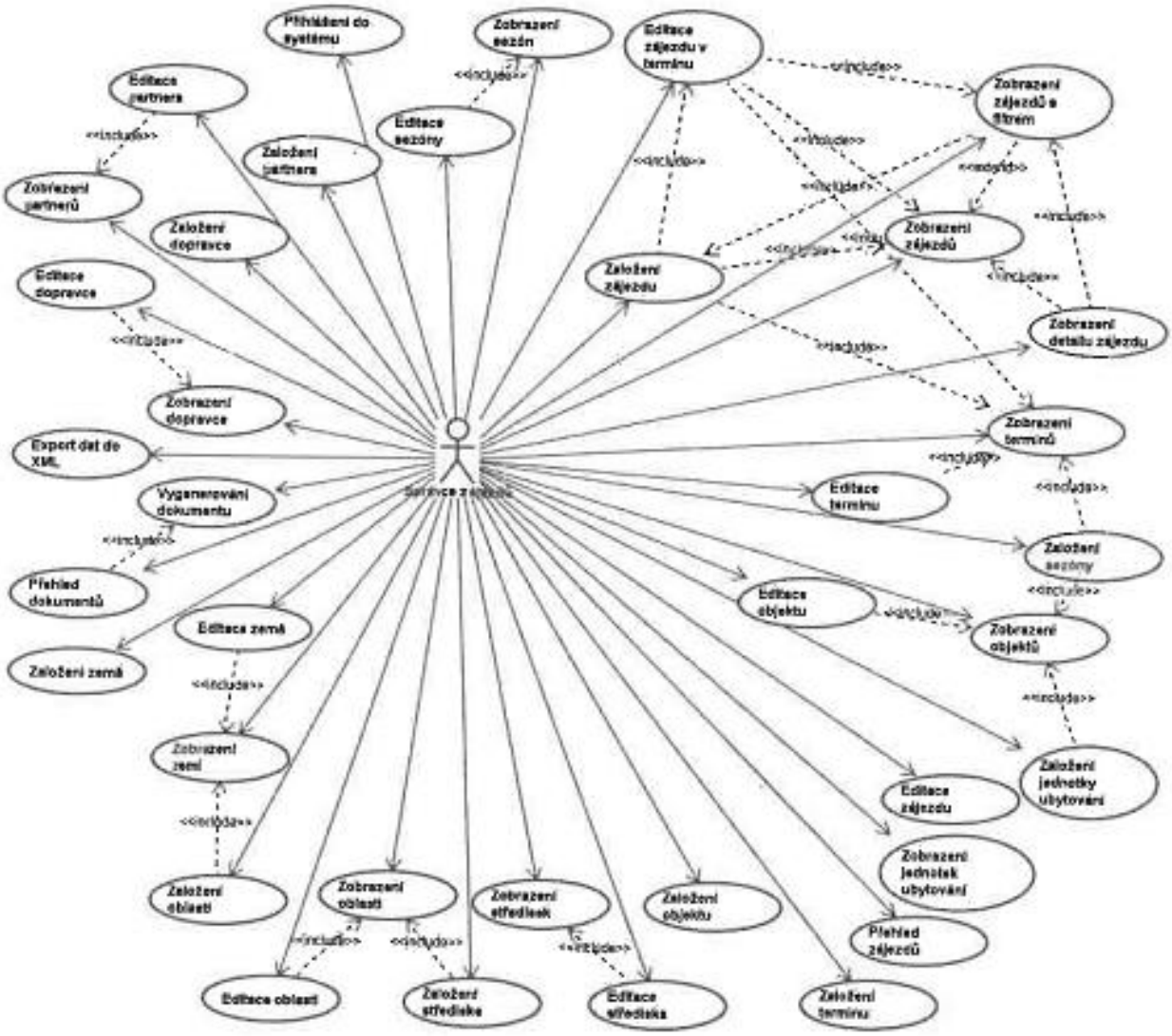
Obrázek 50: USE Case pro aktéra Dealer, zdroj: [BUCH2013] a vlastní úprava

Na obrázku 50 jsou zachyceny případy užití, které realizuje aktér Dealer. Vztah include je zde využit v případu užití Založení rezervace, který zahrnuje Zobrazení detailu zájezdu. Založení zákazníka a Založení cestujícího. V tomto modelu je použit také vztah extend, který „představuje rozšíření základního případu užití. K rozšíření dochází na pojmenovaném místě, které se nazývá bod rozšíření (Extension point), a to jen pokud je splněna podmínka rozšíření. Na rozdíl od vztahu include, základní případ užití u vztahu extend může existovat sám o sobě. Vztah extend použijeme, pokud chceme vytknout podmíněné nebo výjimečné chování anebo přidat chování, které rozšiřuje původní chování. V tomto příkladě případ užití Zobrazení zájezdů s filtrem rozšiřuje Zobrazení zájezdů.



Obrázek 51: USE Case pro aktéra Zákazník, zdroj: [BUCH2013] a vlastní úprava

Na obrázku 51 jsou zachyceny případy užití, které realizuje aktér Zákazník. Jsou to stejné případy užití, které již byly popsány u aktéra Dealer.



Obrázek 52: USE Case pro aktéra Správce zájezdů, zdroj: [BUCH2013]

Na obrázku 52 jsou představeny případy užití, které realizuje aktér Správce zájezdů. Ten provádí činnosti spojené se správou dopravců, partnerů, zemí, oblastí, středisek, objektů, jednotek ubytování, termínů, zájezdů, sezón, zájezdů v termínu.

DETAILNÍ ANALÝZA PRVNÍ ITERACE.

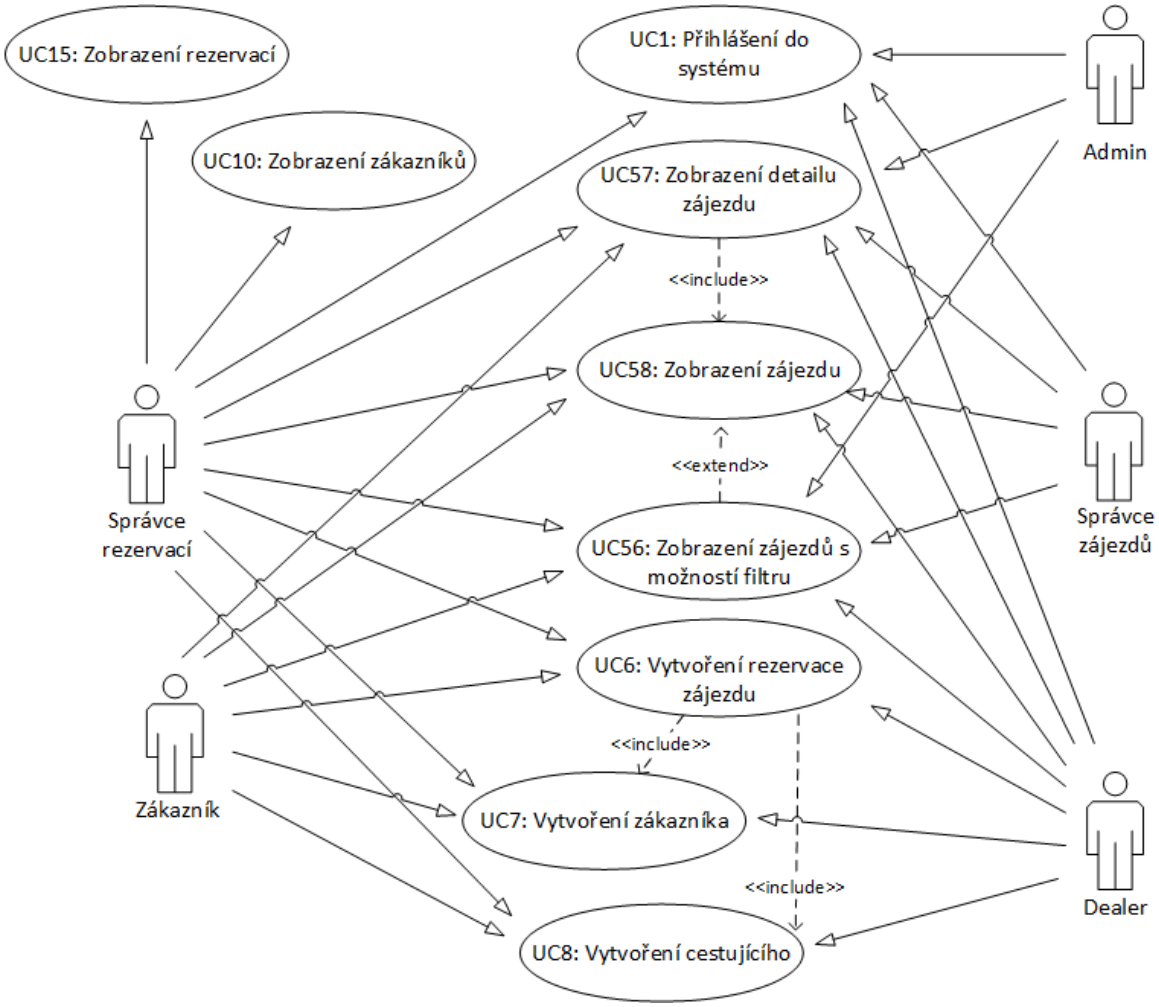
Do první iterace byly vybrány požadavky s prioritou 1. V rámci těchto požadavků byl zpracován Model případů užití, který je zachycen na obrázku 53. Ke každému případu užití byl vytvořen scénář. Jsou uvedeny příklady dvou scénářů.

MODEL PŘÍPADŮ UŽITÍ

V rámci první iterace bylo vybráno devět požadavků, kterým odpovídají případy užití. Případy užití byly doplněny jednoznačným identifikátorem (např. UCl).

V následujícím seznamu jsou stručně charakterizovány jednotlivé případy užití;

- Zobrazení rezervací - zobrazení všech založených rezervací.
- Zobrazení zákazníků - zobrazení všech zákazníků uložených v databázi.
- Přihlášení do systému - přihlášení uživatele systému uživatelským jménem a heslem
- Zobrazení zájezdů - zobrazení všech zájezdů uložených v databázi.
- Zobrazení zájezdů s filtrem - vyhledání zájezdů podle vyhledávácích kritérií, které si uživatel zvolí. Tyto zájezdy se dají třídit podle různých kritérií.
- Zobrazení detailu zájezdu - zobrazení detailu zájezdů, zobrazuje se umístění daného objektu, termíny, ceny, možné příplatky a způsoby dopravy.
- Založení rezervace - založení rezervace pro konkrétní zájezd v termínu. Po úspěšném provedení tohoto procesu se rezervace uloží do databáze.
- Založení zákazníka - tento případ užití je součástí případu užití Založení rezervace a slouží pro definování zákazníka, který zájezd v termínu bude platit.
- Založení cestujícího - v průběhu rezervace se definují cestující, kteří pojedou na zájezd v termínu.



Obrázek 53: USE Case I. Iterace pro všechny aktéry, zdroj: [BUCH2013] a vlastní úprava

SPECIFIKACE PŘÍPADŮ UŽITÍ

Níže jsou uvedeny dva příklady specifikace Use Case, tzv. scénářů případů užití:

- Zobrazení zájezdů s filtrem, tabulka 7
- Založení rezervace, tabulka 8

Název případu užití		**Zobrazení zájezdů s filtrem**	
Identifikace případu užiti		UC1	
Identifikace funkčního požadavku		F011	
Identifikace obrazovek		GUI1	
Byznys pravidla			
Cíl případu užití		Zobrazení zájezdů dle zvolených kritérii a řazení	
Primární aktér(ři)		Správce rezervací, Správce zájezdů, Dealer, Zákazník, Admin	
Pomocný aktér(ři)		-	
Vstupní podmínky		Zákazník na webových stránkách, správce zájezdů a dealer přihlášený do systému, zobrazí se hlavní stránka, kde je možné zvolit kritéria, dle kterých se zobrazí výpis zájezdů	
Výstupní podmínky		Zobrazí se výpis zájezdů dle vybraných kritérií a zvoleného řazení	
Základní scénář	**Krok**	**Role**	**Akce**
	1	**Aktér**	nadefinuje jednotlivá kritéria, dle kterých chce vyhledat zájezdy, může volit z těchto kritérii: Země, Oblast, Středisko, Odjezd od (může vepsat ručně nebo vybrat z kalendáře). Návrat do (může vepsat ručně nebo vybrat z kalendáře). Počet dní (1denni, 2denní, 5denní, 7denní, 9denní, 11denní, 13denní), Typ zájezdu (pobytový, poznávací, lázně a wellness, lyžařský). Doprava (letecky, autobusem, vlakem,). Strava (bez stravy, snídaně, večeře, polopenze, plná penze, all inclusive). Cena od (zadává ručně). Cena do (zadává ručně)
	2	**Aktér**	klikne na tlačítko Vyhledat
	3	**Systém**	zobrazí všechny zájezdy vyhovující zvoleným kritériím dle UC58
	4	**Aktér**	může zvolit, podle čeho chce zobrazené zájezdy řadit (dle ceny vzestupně - přednastaveno defaultně, dle ceny sestupně, data odjezdu vzestupně, data odjezdu sestupně)
	5	**Systém**	seřadí zobrazené zájezdy dle zvoleného řazení
Alternativní scénář – Kritériím nevyhovuje žádný zájezd	**Krok**	**Role**	Akce
	3a 1	**Systém**	zobrazí hlášení: Žádný zájezd nenalezen!
	3a 2	**Systém**	možnost zadání jiných kritérií
	3a 3	**Aktér**	pokračuje dle 1. kroku základního scénáře
Alternativní scénář - Aktér ne nadefinuje kritéria	**Krok**	**Role**	Akce
	1a 1	**Aktér**	klikne na tlačítko Vyhledat
	1a2	**Systém**	zobrazí všechny aktuální zájezdy uložené v DB
	1a 3	**Aktér**	pokračuje dle 4. kroku základního scénáře
Alternativní scénář - Aktér nezvolí řazení	**Krok**	**Role**	Akce
	4a 1	**Aktér**	nezvolí řazeni zájezdů
	4a 2	**Systém**	končí 3. krokem základního scénáře

Tabulka 7: Scénář založení zájezdů s filtrem, zdroj: [BUCH2013]

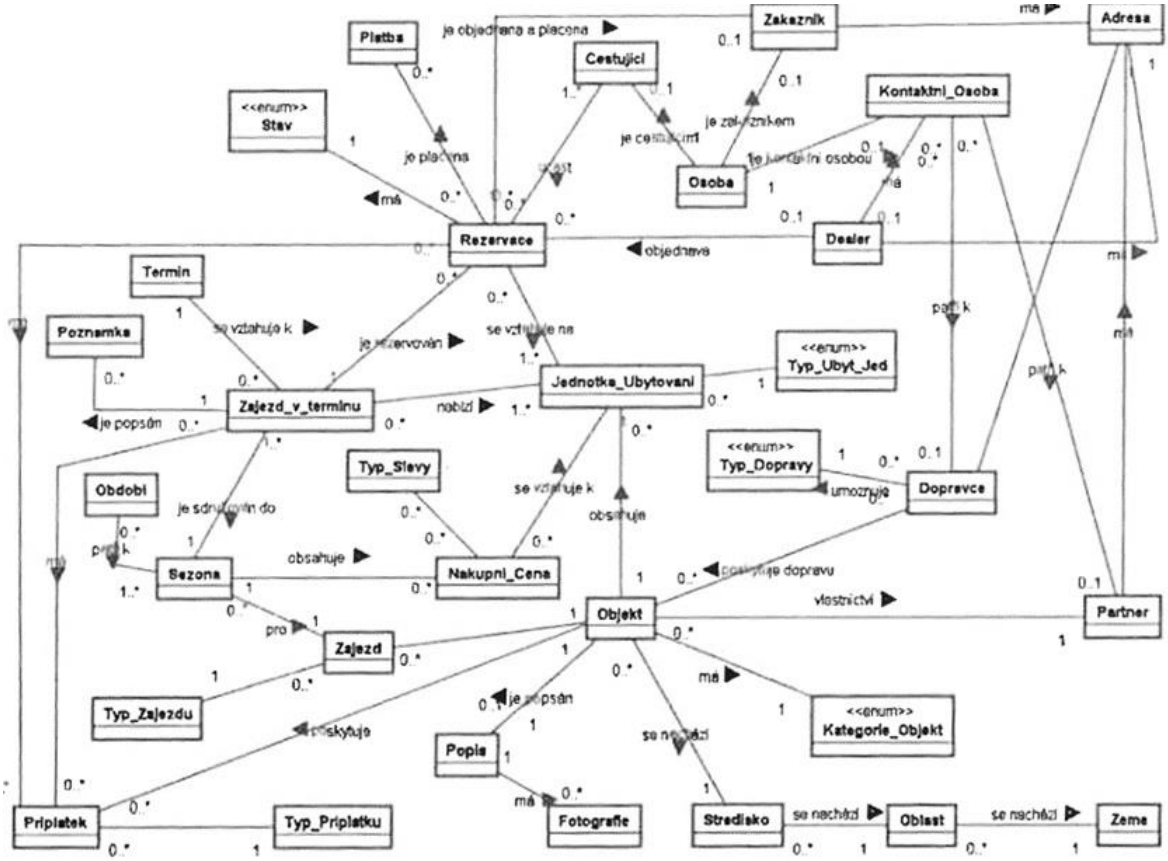
Název případu užiti	**Založeni rezervace**		
Identifikace případu užiti	UC2		
Identifikace funkčního požadavku	F021		
Identifikace obrazovek	OBR_04, OBR_05, OBR_06, OBR_07, OBR_08		
Byznys pravidla			
Cil případu užiti	Vytvoření nové rezervace a její uloženi do DB		
Primární aktér(ři)	Dealer, Správce rezervací, Zákazník		
Pomocný aktér(ří)	-		
Vstupní podmínky	Aktér je přihlášený k aplikaci, je zobrazena hlavní stránka, kde je možné zvolit kritéria, dle kterých se zobrazí výpis zájezdů.		
Výstupní podmínky	Rezervace vytvořena		
Základní scénář	**Krok**	**Role**	**Akce**
	1	Systém	UC57
	2	Systém	zobrazí nové okno pro zakládáni rezervace, nahoře je vypsán Název hotelu. Země, Oblast a Středisko, Typ zájezdu, dále se zobrazí pole: Počet osob a Počet osob do 12 let (podle součtu těchto počtů se u zakládání cestujících zobrazí počet formulářů), Počet dní, Termín (pouze ty, které odpovídají zvolenému počtu dní). Počet jednotek ubytování, Typ jednotky ubytování (dostupné dle počtu jednotek ubytování ve vybraném termínu). Tlačítko pro přidáváni dalšího počtu a typu jednotky ubytování. Strava, Doprava, v dolní části obrazovky Prodejní cena
	3	Aktér	vyplní všechna pole
	4	Systém	zobrazí Prodejní cenu
	5	Aktér	pokud chce zájezd rezervovat, klikne na tlačítko Pokračovat
	6	Systém	zobrazí další okno
	7	Aktér	založí zákazníka **dle UC7**
	8	Aktér	klikne na tlačítko Pokračovat
	9	Systém	zobrazí další okno
	10	Aktér	založí cestující **dle UC8**
	11	Aktér	klikne na tlačítko Pokračovat
	12	Systém	zobrazí další okno se souhrnem rezervace, zobrazují se informace o zájezdu, zákazníkovi a cestujících
	13	Aktér	potvrdí rezervaci kliknutím na tlačítko Odeslat
	14	Systém	zobrazí další okno s hlášením; Rezervace úspěšně odeslána! a tabulkou, kde je zobrazeno; ID rezervace. Výše zálohy. Splatnost, Č. Účtu, Variabilní symbol a hlášení. Informace o Vaší rezervaci byly zaslány na Váš e-mail!
Alternativní scénář - Aktér nechce pokračovat v rezervaci	**Krok**	**Role**	**Akce**
	5a 1	Aktér	pokud se chce vrátit zpět na detail zájezdu, klikne na tlačítko Zpět
	5a 2	Systém	zobrazí stránku s detailem zájezdu
Alternativní scénář- Aktér nechce pokračovat v rezervaci	**Krok**	**Role**	**Akce**
	5b 1	Aktér	pokud chce vyhledávat zájezdy dle jiných kritérií, klikne na tlačítko Nové vyhledávání
	5b 2	Systém	zobrazí hlavní stránku s vyhledáváním zájezdů

Tabulka 8: Scénář založení rezervace, zdroj: [BUCH2013]

DIAGRAMY TŘÍD

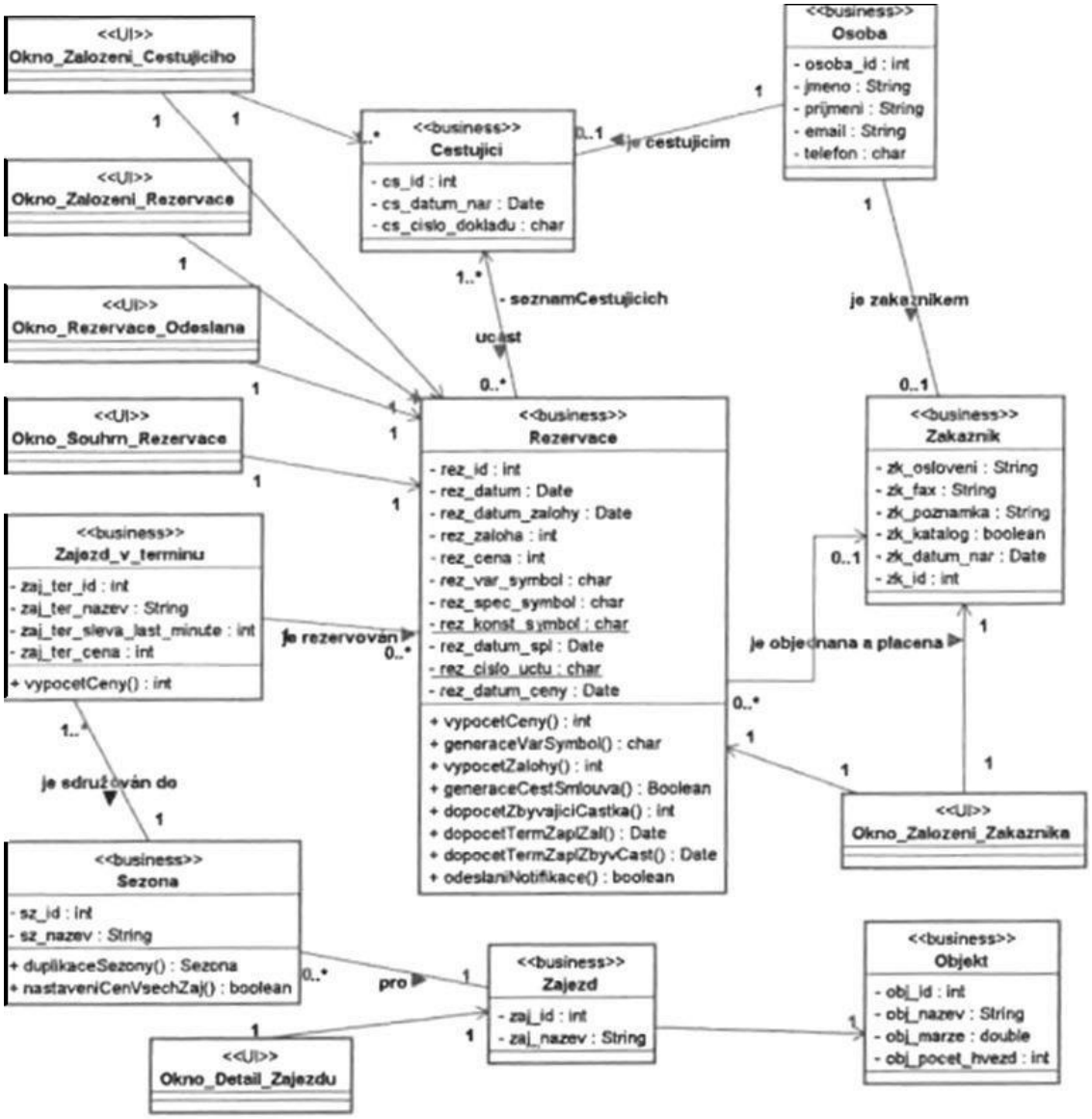
Na obrázku 54 je zachycen **analytický model tříd**, který je součástí dokumentu Popis architektury. Analytický model tříd zachycuje třídy v dané problémové oblasti, tedy byznys třídy a vazby mezi nimi. Protože je model hodně velký, jsou na uvedeném obrázku zachyceny jen názvy tříd bez atributů a metod.

Ústředni třídou modelu je třída Rezervace, se kterou jsou vazbou asociace spojeny další třídy, především Zákazník, Cestující, Zájezd_v_termínu, JednotkaJUbytování, Příplatek, Platba. Asociace jsou pojmenovány, aby bylo jasné, co vazba reprezentuje. Na obou stranách asociace jsou uvedeny násob¬nosti vazby, které vyjadřují, kolik instancí asociované třídy se vazby účastní. Například asociace mezi třídami Rezervace a Zájezd je termínu je pojmenována je rezervován, přičemž vyznačený směr čtení navádí k interpretací vztahu „Zájezd v termínu je rezervován v rezervaci". Násobnosti u této vazby říkají, že v rezervaci je rezervován právě jeden zájezd v termínu a k jednomu zájezdu v termínu může být žádná nebo více rezervací.



Obrázek 54: Analytický model tříd, zdroj: [BUCH2013]

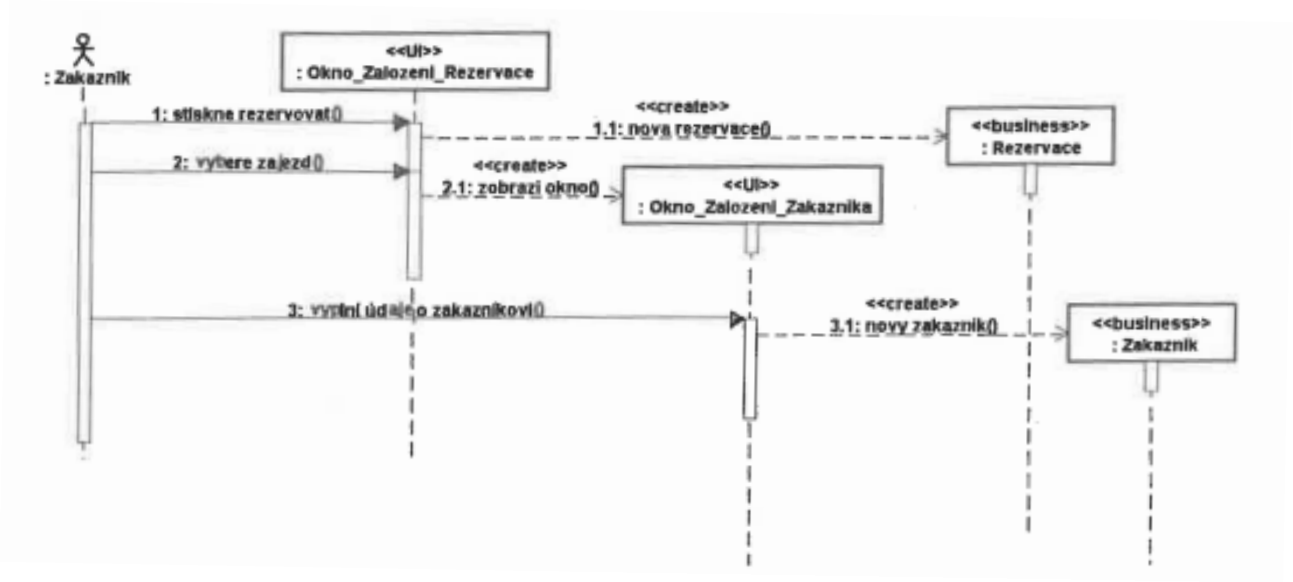
Část designového modelu tříd je zachycena na obrázku 55, kde jsou uvedeny jen některé základní třídy a jim odpovídající třídy (okna) uživatelského rozhraní. Pro rozlišení byznys tříd a tříd uživatelského rozhraní jsou použity u tříd stereotypy business a UI. Třídy byznys logiky mají uvedeny atributy a metody. U atributů je uvedena viditelnost, datové typy, u metod návratové hodnoty a parametry. Vazby mezi třídami jsou navigovatelné.



Obrázek 55: Designový model tříd, zdroj: [BUCH2013]

SEKVENČNÍ DIAGRAM

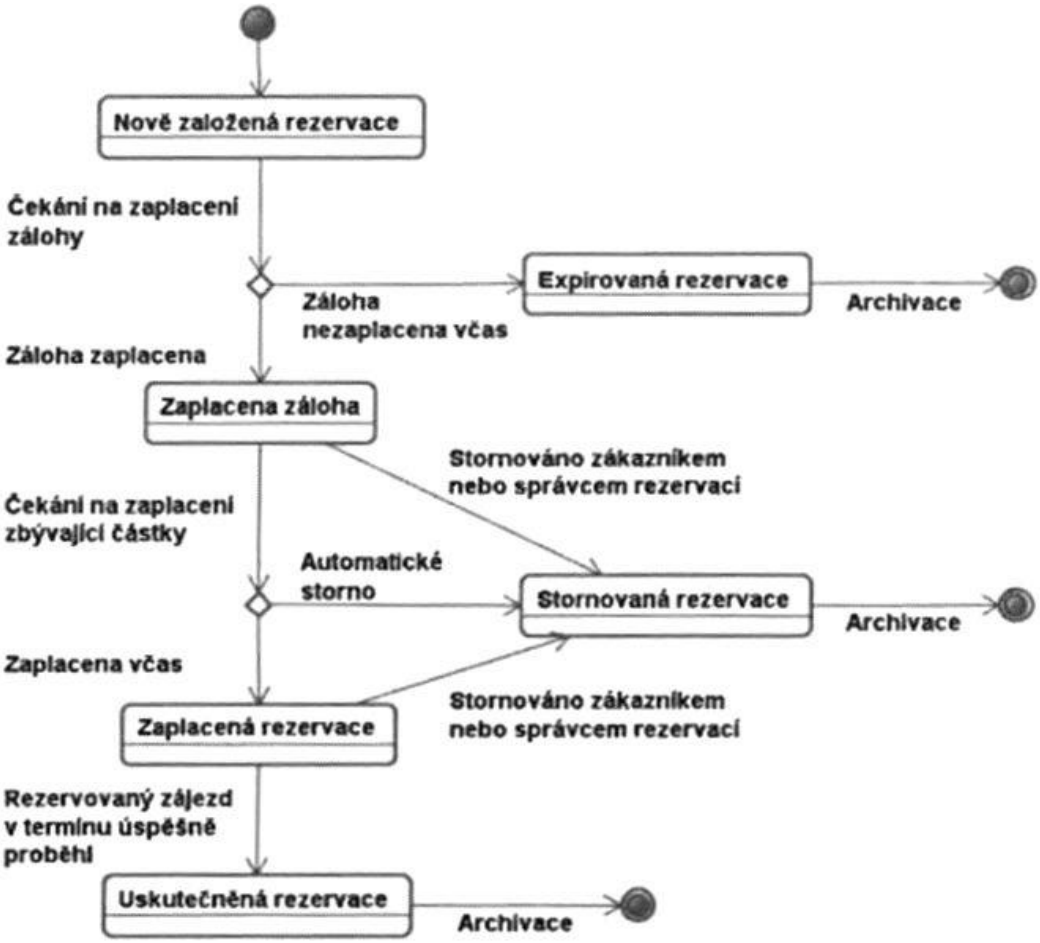
Na obrázku 56 je zachycena malá část sekvenčního diagramu, znázorňující komunikaci mezi aktérem Zákazník a aplikací, tedy třídou Okno_Založení_Rezervace, která na základě zadaných údajů vytvoří objekt třídy Rezervace a přejde na obrazovku Založení_zákaznika, která vytváří objekt třídy Zákazník.



Obrázek 56: Sekvenční diagram, zdroj: [BUCH2013]

STAVOVÝ DIAGRAM

Na obrázku 57 je zachycen stavový diagram pro objekt Rezervace znázorňující stavy Nově založená rezervace, Expirovaná rezervace. Zaplacená záloha. Zaplacená rezervace. Stornovaná rezervace a Uskutečněná rezervace. Nakonec se rezervace archivuje. Změna stavů probíhá na základě událostí.



Obrázek 57: Stavový diagram, zdroj: [BUCH2013]

SHRNUTÍ STUDIJNÍ OPORY

Studijní opora „Objektové metody modelování v příkladech“ je určena studentům studijního programu manažerská informatika na Obchodně podnikatelské fakultě v Karviné, Slezské univerzity v Opavě.

Vznikla na základě zkušeností autora s vedením seminářů a přednášek v předmětu Objektové metody modelování.

Studijní opora ve spojení poskytuje ucelený náhled do problematiky analytického procesu návrhu informačního systému, tvorby software nebo dokumentace software. Postihuje fázi návrh systému, analýzy systému předcházející zahájení programátorských prací.

Velký důraz je kladen při výkladu látky na příklady z praxe, které provázejí celý text a je jim věnována samostatná závěrečná kapitola.

Problematika objektových metod modelování, návrh informačních systémů a tvorba software je velmi rozsáhlá. Pokrývá celý vývojový proces od návrhu informačního systému, přes jeho programování až po jeho nasazení, údržbu a inovování. Tato studijní opora je věnována zejména části analýzy, která předchází programování.

Popsané poznatky a přístupy při návrhu informačních systémů jsou nezbytným vědomostním vybavení studenta studijního programu „Manažerská informatika“ Slezské univerzity v Opavě, Obchodně podnikatelské fakultě v Karviné.

LITERATURA

[1] [Arl2008] ARLOW, J. NEUSTADT, I. UML 2 a unifikovaný proces vývoje aplikací: Objektově orientovaná analýza a návrh prakticky. Přel. Bogdan Kiszka. Dotisk 1.vydání, Brno: Computer Press, 2008. 567 s. ISBN 978-80-251- 1503-9.

[2] [Ald2005] ALDORF, F. Metodika RUP - diplomová práce, 2005, VŠE Praha

[3] [Boo1998] BOOCH, G. JACOBSON, I. RUMBAUGH, J. The Unified Modeling Language User Guide. 1.vyd. Addison Wesley, 1998. ISBN 0- 201-57168-4.

[4] [Buch2007] BUCHALCEVOVÁ, A. PAVLÍČKOVÁ, J. PAVLÍČEK, L. Základy softwarového inženýrství - materiály ke cvičení. 1.vyd. Praha: Vysoká škola ekonomická, 2007. 222 s. ISBN 987-80-245-1270-9.

[5] [Fow2003] FOWLER, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3. edition. Addison Wesley, 2003. 208 s. ISBN 0- 321-19368-7.

[6] [Kan2004] KANISOVÁ, H. Miller, M. UML srozumitelně, 1.vyd, Brno: Computer Press, 2004. 158 s. ISBN 80-251-0231-9.

[7] [Pen2003] PENDER, Tom. UML Bible. 1.vyd. Indianapolis: Wiley Publishing, Inc., 2003. ISBN 0-7645-2604-9.

[8] [Rum2004] RUMBAUGH, J. JACOBSON, I. BOOCH, G. The Unified Modeling Language Reference Manual. 2.vyd. Addison-Wesley, 2004. ISBN 032124562853

[9] [Schm2001] SCHMULLER, J. Myslíme v jazyku UML : knihovna programátora. přel. Jiří Hynek. 1.vyd. Praha: Grada, 2001. 360 s. ISBN 80-247-0029-8.

[10] [Str2007] STŘÍŽOVÁ, V. HORNÝ, S. SVATÁ, V., VÁCLAVÍKOVÁ, M. Systémové pojetí (hospodářské) organizace. 1.vyd. Praha: Oeconomica, 2007. 239 s. ISBN 978-80-245-1265-5.

[11] [Lef2010] LEFFINGWELL, D. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise***.** Addison-Wesley Professional, 2010, 560 s., ISBN 0-321-63584-1

[12] [Fow2009] FOWLER, M**.** Destilované UML**.** Grada Publishing a.s. 2009. 176 s. ISBN 978-80-247-2062-3

[13] [Fra2014] FRANĚK Z. Objektové metody modelování, učební text, SU OPF 2014, 115 s., elektronická verze „online“, ISBN 978-80-7510-081-8

[14] [BUCH2013] BUCHALCEVOVÁ, A., STANOVSKÁ I. Příklady modelů analýzy a návrhu aplikace v UML, Vysoká škola ekonomická v Praze, Nakladatelství Oeconomica, 2013, 198 s., ISBN 978-80-245-1922-7

Internetové zdroje:

[1] http://mpavus.wz.cz

[2] http://uml.czweb.org/index.html

[3] http://www.rational.com

[4] http://www.ibm.com

[5] http://www.sparx.com

[6] http://ecom.ef.jcu.cz/web2/download/podklady/zakladni-pojmy-ea.pdf

[7] ttp://dspace.upce.cz/bitstream/handle/10195/64679/E1.pdf?sequence=1&isAllowed=y

[8] https://www.root.cz/clanky/nastroje-pro-tvorbu-uml-diagramu/

PŘÍLOHA Č. 1: SEZNAM OBRÁZKŮ

PŘÍLOHA Č. 2: SEZNAM TABULEK

PŘEHLED DOSTUPNÝCH IKON


Čas potřebný ke studiu


Cíle kapitoly


Klíčová slova


Nezapomeňte na odpočinek


Průvodce studiem


Průvodce textem


Rychlý náhled


Shrnutí


Tutoriály

Definice


K zapamatování


Případová studie


Řešená úloha

Věta


Kontrolní otázka


Korespondenční úkol


Odpovědi


Otázky


Samostatný úkol


Další zdroje


Pro zájemce


Úkol k zamyšlení

Název: **Objektové metody modelování v příkladech**

Autor: **Tomáš Barčák RNDr., Zdeněk Franěk, Ph.D.**

Vydavatel: Slezská univerzita v Opavě

Obchodně podnikatelská fakulta v Karviné

Určeno: studentům SU OPF Karviná

Počet stran: 132