UNIVERZITA PARDUBICE
FAKULTA EKONOMICKO-SPRÁVNÍ

BAKALÁŘSKÁ PRÁCE

2011 **Michal VÁVRA**

UNIVERZITA PARDUBICE
FAKULTA EKONOMICKO-SPRÁVNÍ

Case nástroj ObjectiF (demo) pro objektovou analýzu - výukové příklady

Michal Vávra

Bakalářská práce

2011

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal VÁVRA**
Osobní číslo:	**E08991**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Case nástroj ObjectiF (demo) pro objektovou analýzu - výukové příklady**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

Cílem bakalářské práce je vytvoření příkladů, které využijí studenti prezenčního i kombinovaného studia. Příklady jsou členěny na analýzu, návrh a implementaci v prostředí SW nástroje ObjectiF (demo verze).

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

MULLER, Robert. *Database Design for Smarties: Using UML for Data Modeling*. **San Francisco: Morgan Kaufmann Publisdhers, 1999. 464s. ISBN 1-55860-515-0.**

PENDER, Tom. *UML Bible*. **Wiley. 2003. 984s. ISBN 978-0-7645-2604-6.**

TEOREY, Toby, LIGHTSTONE, Sam, NADEAU, Tom. *Database Modeling and Design*. **Morgan Kaufmann Publisdhers, , 2005. 296 s. ISBN 9780126853520**

Vedoucí bakalářské práce: **Ing. Stanislava Šimonová, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání bakalářské práce: **29. září 2010**

Termín odevzdání bakalářské práce: **6. května 2011**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Křupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 29. září 2010

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na mojí práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména na skutečnosti, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 20.6.2011 Michal Vávra

Poděkování

Chtěl bych poděkovat vedoucí mé bakalářské práce paní Ing. Stanislavě Šimonové, Ph.D., za rady, které mi poskytla v průběhu zpracovávání této bakalářské práce.

ANOTACE

Tato práce obsahuje analýzu informačního systému firmy pomocí programu objectiF, návrh informačního systému pro relační databázi a implementaci v programu Access. Práce je rozdělena na tři části. První část je pojata teoreticky, jsou v ní popsány přístupy k analýze a návrhu informačního systému. Dále je v této části popsán jazyk UML s diagramy použitými v praktické části práce. Druhá část je zaměřena na volbu skladby tématik, volbu modelovaných nástrojů, volbu modelovaných reálií a volbu struktury řešení praktické části. V třetí části je popsána modelovaná reálie s diagramy, které zachycují elementy a vztahy vymezené v druhé části práce. Ucelená linie příkladů zachycující analýzu, návrh a implementaci je v příloze práce.

KLÍČOVÁ SLOVA

UML, use case diagram, hlavní scénář, alternativní scénář, sekvenční diagram, diagram tříd, modelování, analýza informačního systému, návrh informačního systému, CASE nástroje.

ANNOTATION

This work includes analysis of company information system using objectiF, concept of information system for a relational database and implementation using Access. The work is divided into three parts. The first part is conceived theoretically, there are described approaches to the analysis and concept of information systems. In addition, this part describes the UML diagrams that were used in the practical part. The second part focuses on the choice of item structure, choice of the modeled instruments, choice of modeled realia and choice of practical part structure. In the third part is described modeled realia with diagrams that capture an elements and relationships defined in the second part of the work. Complete line of examples intercepting analysis, concept and implementation are in the annex.

KEYWORDS

UML use case diagram, main scenario, alternative scenario, sequence diagram, class diagram, modeling, analysis of information system, concept of information system, CASE tools.

OBSAH

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

ASCII	American Standard Code for Information Interchange
CASE	Computer Aided Software Engineering
DFD	Data Flow Diagram
ERD	Entity Realationship Diagram
FSD	Function Structure Diagram
ID	Jedinečný identifikátor
IS	Informační systém
STD	State Transition Diagram
UML	Unified Modeling Language

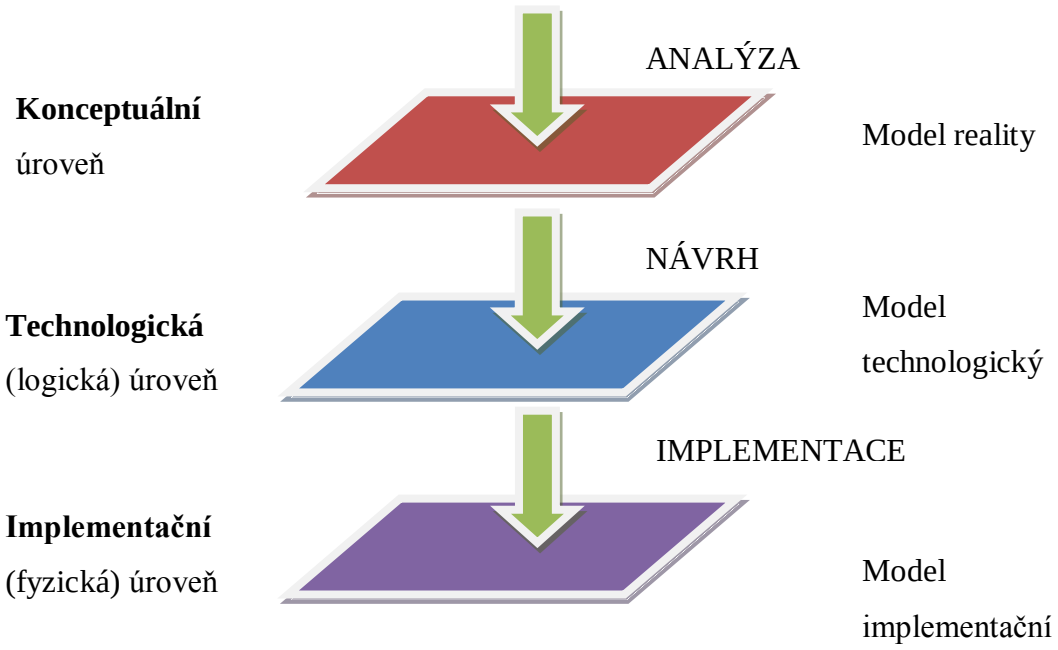
1 Úvod

V této práci jsem se zaměřil na tématiku analýzy, návrhu a implementace informačních systémů. V dnešní době využívají firmy informační systémy (IS) k tomu, aby podporovaly jejich podnikatelskou činnost. V průběhu času se IS musí přizpůsobovat měnícím se podmínkám (legislativa, změna chování zákazníků, změny u dodavatelů). IS umožňuje pracovníkům automatizovat jejich každodenní úkony a dále pak také umožňuje manažerům sledovat výkonnost organizace. IS slouží především k uchování, zpracování a prezentaci dat. K vytvoření IS je nutné, aby se zaměstnanci spolupodíleli na analýze i návrhu, jelikož je žádoucí, aby jejich každodenní úkony byly podporovány vytvářeným IS, který pak tyto úkony do nejvyšší míry automatizuje. K vytvoření a zobrazení modelu IS slouží nástroje typu CASE (Computer Aided Software Engineering), mezi které patří např. program objectiF (Microtool).

Cílem bakalářské práce je vytvoření příkladů, které využijí studenti prezenčního i kombinovaného studia. Příklady jsou členěny na analýzu, návrh a implementaci v prostředí SW nástroje objectiF (demo verze).

2 Analýza, návrh a implementace informačních systémů

V průběhu vývoje prostředků pro analýzu a návrh IS se vyprofilovaly dva základní přístupy – přístup strukturovaný a přístup objektově orientovaný. Jednou ze základních technik používaných při analýze a návrhu je modelování. Při modelování jde o to, zobrazit daný systém vhodnou zobrazovací metodou, z různých úhlů pohledu, na různých úrovních podrobnosti. [3]



Obrázek 1: Koncept tří úrovní (zdroj: vlastní – zpracováno na základě [11])

Obrázek zachycuje princip tří úrovní tvorby informačního systému. Každé úrovni odpovídají určité nástroje modelování a metody vývoje. Na každé úrovni se zároveň řeší jiné problémy.

Rozeznáváme datové modely:

- Konceptuální model: představuje popis obsahu systému, úroveň je nezávislá na vlastním implementačním a technologickém prostředí,
- Technologický model: představuje popis způsobu realizace systému v termínech jisté třídy technologického prostředí,
- Implementační model: představuje popis vlastní realizace systému v konkrétním implementačním prostředí. (např. MS Access)

Výstupem konceptuální úrovně je konceptuální model. Tento model vstupuje do další fáze datového modelování – technologické úrovně. V této fázi je nutné se rozhodnout, v jakém typu software bude informační systém vytvořen (např. hierarchická nebo relační databáze). Výstupem technologické úrovně je model, který je orientován na uvažovanou softwarovou platformu. Tento model je vzorem pro vlastní fyzickou implementaci. [11]

Relační model – normalizace dat

Při návrhu relační databáze se uplatňuje přístup shora dolů, tj. transformace konceptuálního modelu do relačního modelu dat.

Odstranění anomálií v datovém modelu je snahou procesu zvaného normalizace dat. Důsledkem normalizace dat je postupná dekompozice datového modelu rozdělením atributů do většího počtu relací, které již nevykazují dané nedostatky. Postupně je množina všech relací převáděna do tzv. vyšších normálních forem

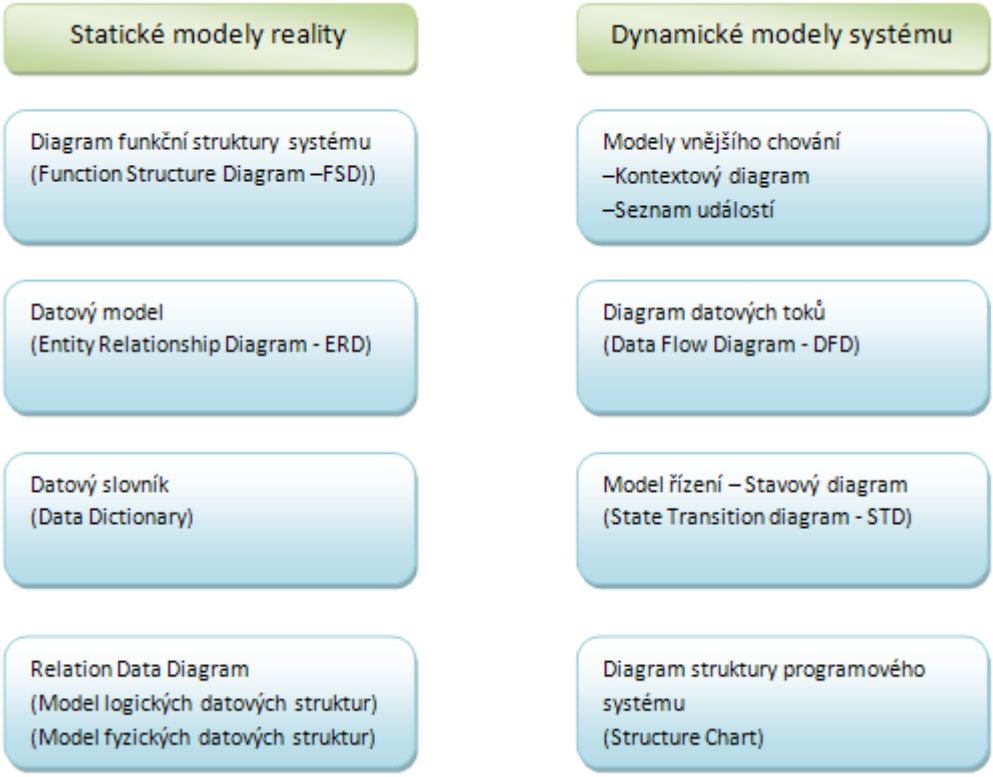
Základní normální formy:

- 1NF (první normální forma): 1NF splňuje relace, která neobsahuje vícehodnotové atributy.
- 2NF: splňuje relace, je-li v 1NF a každý neklíčový atribut je plně funkčně závislý na primárním klíči relace. Týká se pouze více atributového primárního klíče. Neklíčový atribut musí být závislý na celém primárním klíči, nikoliv pouze na jeho části.
- 3NF: splňuje relace, jestliže je v 2NF a každý neklíčový atribut je netranzitivně závislý na primárním klíči (atribut nezávisí na jiném atributu závisejícím na primárním klíči). [11]

2.1 Strukturovaný přístup k analýze a návrhu

Strukturovaný přístup je vytvořený přímo pro potřeby tvorby databáze, resp. je cílený k návrhu a implementaci informačního systému v rámci databázového systému. Analýza je tímto přístupem je realizována s využitím diagramu ERD (Entity Realationship Diagram). ERD zachycuje základní datové objekty, tzv. entity (angl. entities) a vztahy mezi entitami (angl. relationships). [9]

V průběhu analýzy a návrhu je třeba zobrazit dva hlavní aspekty vyvíjeného IS – procesy probíhající v systému a data, se kterými systém pracuje. V této kapitole jsou charakterizovány modely používané při strukturované analýze a návrhu IS. Popisované modely jsou schematicky zobrazeny – viz Obrázek 2.



Obrázek 2: Schéma modelů strukturované analýzy a návrhu IS (zdroj: vlastní - zpracováno na základě [3])

Kontextový diagram

Kontextový diagram je výchozím modelem při modelování IS. Jeho účelem je zachytit veškerou komunikaci (datové toky) mezi vyvíjeným systémem a jeho okolím, rovněž tak zachytit všechny s vyvíjeným systémem komunikující entity. Jde o model příslušející konceptuální vrstvě zobrazení vyvíjeného IS. Lze také říci, že kontextový diagram je speciálním případem diagramu datových toků. [3]

Seznam událostí

Seznam událostí je opět modelem znázorňujícím vnější chování systému, obdobně jako kontextový diagram. Je však zaměřen speciálně na datové toky vstupující do vyvíjeného systému z okolí, které mají charakter podnětů k vyvolání reakce systému, k nastartování dalších návazných procesů a datových toků. [3]

Diagram funkční struktury systému

Diagram funkční struktury systému nahlíží na systém jako na celek, který se dělí na další subsystémy až do potřebné úrovně podrobnosti. Jako prostředek pro vyjádření organizační struktury podniku a výroby. [3]

Diagram datových toků

Diagram datových toků nevyjadřuje časové uspořádání procesů ani datových toků. Znázorněná návaznost mezi procesy a datovými toky v DFD je čistě datová (data do procesů vstupují a z procesů vystupují). Chybí zde znázornění návaznosti mezi datovými toky a procesy v závislosti na čase a na pořadí datových toků procesů. [3]

Datový model

Datový model zobrazuje strukturu dat vyvíjeného IS. ERD znázorňují podrobnou strukturu úložišť dat, které byly zobrazeny v diagramech datových toků.

Mezi základní pojmy datového modelu patří:

- Entita – pod pojmem entita rozumíme každý objekt, o kterém má být uchovávaná informace ve vyvíjeném IS. Musí jít o rozlišitelný a identifikovatelný objekt reality.
- Atribut entity – atributy jsou prvky entity, které slouží k popisu vlastností entity (např. jméno osoby)
- Primární klíč – je to atribut, jehož hodnota slouží k jednoznačné identifikaci jednotlivých entit. [3]

Datový slovník

Datový slovník je modelem, ve kterém pro zobrazení vyvíjeného IS používáme jazyk slovního popisu. Datový slovník musí obsahovat popis prvků uvedených v jednotlivých modulech. Popis musí být přehledný, srozumitelný. Musí zachycovat i vazby mezi popisovanými prvky. Účelem datového slovníku je popis všech prvků systému vyskytujících se ve zpracovaných modulech. Jde o významový slovník všech identifikátorů a spojení mezi prvky modelu. [3]

Stavový diagram

Stavové diagramy mají význam zejména při modelování systémů pracujících v reálném čase. Účelem stavového diagramu je modelování časově závislého chování systému. Stavový diagram zachycuje chování systému, jeho reakce na vnější nebo vnitřní události. [3]

Diagram struktury programového systému

Účelem diagramu struktury programového systému je znázornění hierarchie závislosti modulů programového systému, ze kterých se softwarové vybavení informačního systému bude skládat. Jde o model, který se vytváří v etapě detailního návrhu IS a který může být dále upřesňován v etapě implementace. [3]

Relation data diagram

Relační model vychází ze souboru základních matematických principů odvozených z teorie množin a predikátové logiky. Relační model definuje způsob, jakým je možné data reprezentovat, způsoby jejich ochrany a dále operace, které můžeme nad daty provádět. [11]

2.2 Objektově orientovaný přístup k analýze a návrhu – UML

Modelovací jazyk, označovaný zkratkou UML, je jazyk, který má vlastní grafickou syntaxi (tj. pravidla pro sestavování jednotlivých elementů jazyka do větších objektů) a sémantiku (tj. jednoznačná pravidla určující jednotlivým syntaktickým výrazům jejich význam). [13]

V současné době má jazyk UML velký význam při návrhu softwarových systémů, protože objektově orientovaný návrh každé složitější aplikace je nezbytným předpokladem pro její úspěšnou a rychlou implementaci. Pro objektově orientovaný návrh je možné použít různé podpůrné prostředky, zejména další odlišné typy diagramů. UML je však významné také v tom ohledu, že přesně specifikuje, co má daný diagram obsahovat, což je velmi důležité zejména při sdílení informací mezi jednotlivými analytiky a vývojáři. Dále je již z principu UML nutné, aby vytvářené diagramy měly vnitřní konzistenci a přesně danou sémantiku, což u jiných typů diagramů nemusí být

obecně zaručeno. UML diagramů existuje několik typů lišících podle toho, jaké úlohy se pomocí nich plánují či zpracovávají. Tyto diagramy se od sebe odlišují především repertoárem použitých značek, způsobem jejich vzájemného propojení a s nimi související sémantikou. [13]

Mezi přednosti jazyka UML i na něm postavených UML diagramů patří existence otevřeného a rozšiřitelného standardu, podpora celého vývojového cyklu aplikace či jiného (ne nutně programového) systému a podpora pro různé aplikační oblasti. Pro širší využití jazyka UML v praxi mluví také fakt, že je podporován celou řadou vývojových nástrojů, ať už samostatných aplikací určených pouze pro práci s UML, nebo i integrovaných vývojových prostředí, které v některých případech dovolují provádět převod informací mezi UML diagramem a algoritmem zapsaným v programovacím jazyce (a samozřejmě i opačným směrem, ten je však z pochopitelných důvodů složitější a ne vždy uskutečnitelný). [13]

UML není ucelená metodika definující proces tvorby IS, ale množina nástrojů, které se v tomto procesu používají. Teprve nasazením odpovídající objektové metodiky, která definuje kdy, kým a jak jednotlivé nástroje používat, můžeme vytěžit z UML maximum. UML odlišuje nástroje pro modelování statické a dynamické stránky systému. Zjednodušeně řečeno, statický pohled ukazuje, jak systém vypadá, zatímco dynamický definuje, jak se systém chová v čase a jak reaguje na nejrůznější podněty. Souhrnně hovoříme o architektuře systému. Jednotlivé nástroje a prostředky obsažené v UML nám při respektování tohoto pravidla dovolí modelovat vytvářený systém ze všech možných stran a na všech možných úrovních. Vhodnou kombinací těchto nástrojů lze dosáhnout přehledných modelů dílčí i celkové architektury systému. [15]

Prvním z úkolů analýzy je vymezení hranic modelovaného systému. Tento postup je samozřejmě analogický s procesem zkoumání jakéhokoliv předmětu. Zpravidla první a určující činností je vždy tento zkoumaný předmět poznat, čímž se zabývá obecně analýza, a následně se dají na základě těchto poznatků dělat další rozhodnutí, která mohou mít naopak povahu syntézy (etapy designu a implementace). Po úspěšném vymezení hranic systému následuje nalezení vhodných objektů a vztahů mezi nimi, které tvoří danou oblast. Takto vytvořený objektový model je poté různě zpracováván až do podoby předloh pro implementaci, pro kterou lze využít automatické generování kódu příslušného CASE nástroj. [15]

2.2.1 Notace a diagramy

Základem definice syntaxe UML je specifikace lexikálních elementů, tzn. nejmenších jednotek, ze kterých se může dokumentace v UML skládat. Z lexikálních elementů lze sestavovat různé diagramy. [10]

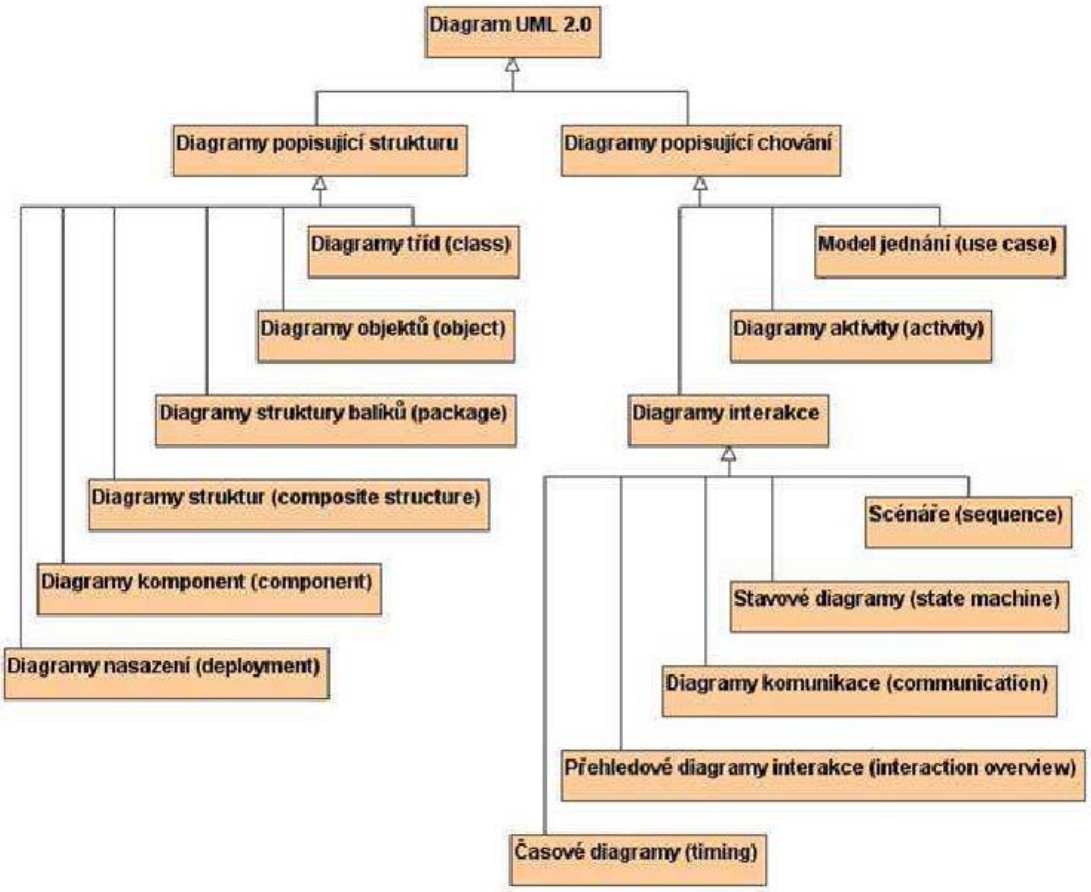
UML definuje následující lexikální elementy:

- Řetězec, tj. posloupnost znaků – znaková sada není omezena, avšak je vhodné používat kód ASCII
- Ikona, tj. dále nedělitelný grafický symbol zastupující nějaký element. V ikonickém zobrazení nejsou zobrazeny žádné složky elementu, což ovšem neznamená, že žádné složky nemůže mít.
- 2D-symbol, tj. grafický symbol zastupující složený element. 2D-symbol je rozdělen na oblasti vystihující různé aspekty popisu elementu. Obsah oblasti představuje složky zobrazeného elementu.
- Spojnice (path), tj. čára, která může mít zvýrazněny koncové body na hranici ikon nebo 2-D symbolů sloužící k vyznačení propojení odpovídajících elementů. Takové propojení vyjadřuje jistou souvislost mezi elementy – může to být souvislost symetrická, nebo nesymetrická. V případě symetrických souvislostí se používají orientované spojnice – jeden koncový bod spojnice je označen šipkou znázorňující směr orientace. [10]

Složky modelů / diagramů obsahují:

- Prvky: objekty, třídy, struktura, chování,
- Vazba: mezi prvky,
- Ostatní: dekorace, poznámky a dokumentace. [10]

UML diagramy se člení do této struktury diagramů (Obrázek 3). Blíže budou popsány pouze diagramy použité v praktické části bakalářské práce.



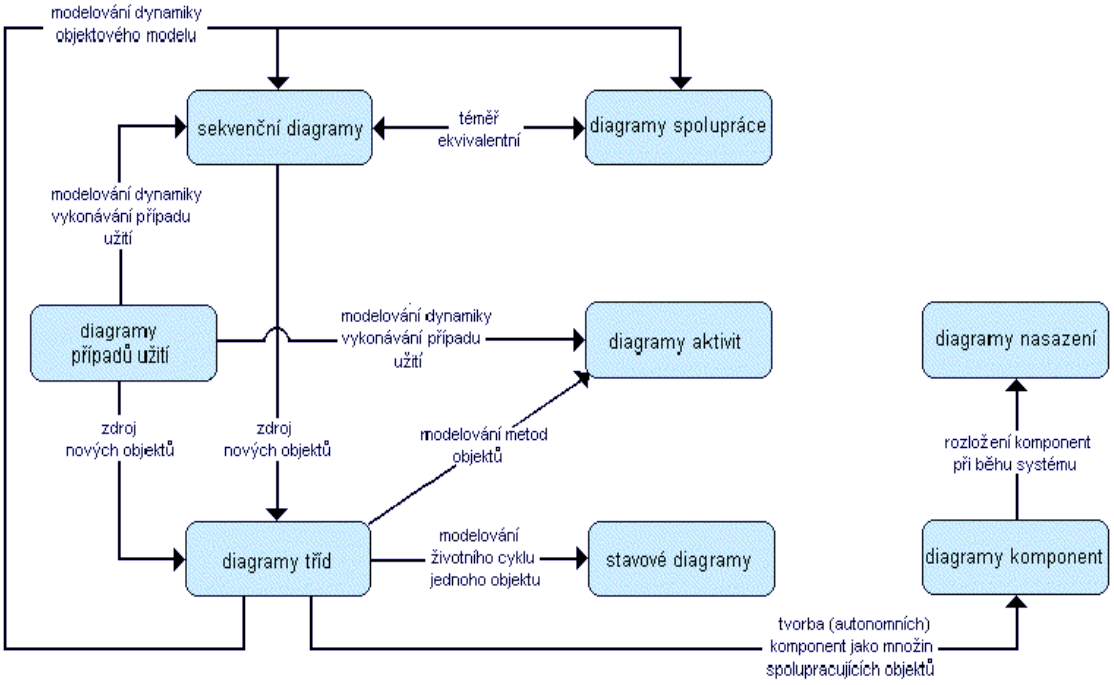
Obrázek 3: Přehled digramů UML (zdroj:[1])

Model v UML se skládá z různých diagramů, jež představují průhledy na různé části sémantického základu navrhované aplikace. Sémantickým základem je souhrn specifikací aplikace, který vymezuje teritorium, v němž se může návrh pohybovat. Diagram ve vizuální formě zobrazuje právě jeden konkrétní pohled na aplikaci. Žádný dvourozměrný diagram nemůže zachytit komplexní aplikaci v celku, ale soustředí se vždy právě na jeden důležitý aspekt. Jazyk UML rozeznává pět základních pohledů na systém. [8]

- *Pohled případů užití.* V případech užití jsou vyjádřeny základní požadavky kladené na systém. Veškeré další pohledy se pohybují v mantinelech vymezených pohledem případů užití.
- *Logický pohled* se zabývá zejména pojmy z problémové domény zadavatele a jejich vzájemnými statickými vztahy.

- *Procesní pohled* se soustřeďuje na chování systému, které musí splňovat požadavky a omezení z případů užití, jež jsou kladeny na průběh procesů. Pregnantně řečeno, jedná se o procesně orientovaný doplněk logického pohledu.
- *Implementační pohled* zachycuje fyzické rozdělení aplikace na samostatné komponenty a jejich závislosti.
- *Pohled nasazení* mapuje komponenty na množinu fyzických výpočetních uzlů v cílovém prostředí. [8]

Na následujícím obrázku (Obrázek 4) jsou vidět vztahy mezi základními diagramy, které umožňuje UML zobrazit.



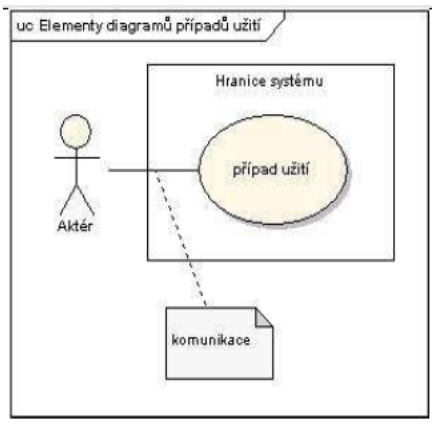
Obrázek 4: Vztahy mezi základními diagramy UML (zdroj:[2])

2.2.2 Diagram případů užití

Základním pohledem dokumentujícím funkčnost systému je model jednání. Model jednání zahrnuje diagramy případů užití a jejich popis. Přehled základních elementů diagramů případů užití obsahuje Tabulka 1. [1]

Diagram případů užití se vytváří s pomocí uživatelů systému pro zjištění funkčních požadavků na systém. Na základě rozhovorů s uživateli se zjišťují a zapisují případy užití. Počáteční rozhovory slouží k odhalení existence aktérů a případů užití vyšší úrovně, které popisují funkční požadavky na systém, hranice a rozsah systému. Diagram případů užití představuje soubor případů užití, aktérů a jejich vzájemných vztahů. Každý případ užití popisuje posloupnost událostí. Každá posloupnost je inicializovaná jistou entitou, která se nazývá aktér. [7]

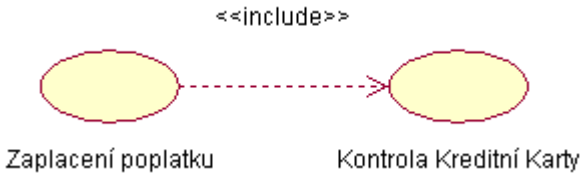
Tabulka 1:Elementy diagramů případů užití (zdroj:[1])

Element	UML sémantika	UML syntaxe
Aktér	znázorňuje uživatelskou roli, spolupracující výpočetní systém či čas	
Hranice systému	vyznačuje oblast zodpovědnosti popisovaného systému	
Relace	Vyznačuje vztah mezi jednotlivými případy užití	
Případ užití	dokumentuje možnost využít systém k realizaci služby	
Komunikace	vyznačuje, který aktér může danou událost vyvolat – představuje komunikační kanál mezi aktérem a systémem	

Typy relací v use case diagramech

- ***Include (zahrnutí)***

Tato relace se objevuje tam, kde existuje podobná nebo stejná část sekvence scénáře, opakující se ve více případech užití. V takovém případě není vhodné udržovat více kopií shodných části scénářů, ale doporučuje se vyčlenit samostatný případ užití, obsahující opakující se část scénáře. Jinými slovy, jedná se o vyčlenění společného chování ze scénářů základních případů užití. Použitému případu užití je jedno, kdo jej používá. Dodává svoje chování do základního případu užití, který pro tento účel musí definovat přesný bod, ve kterém je řízení předáno do rozšiřujícího případu užití (podobně jako volání funkce). Podstatné je, že základní případ užití není bez rozšiřujícího případu užití kompletní. [4]

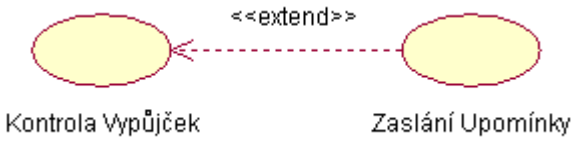


Obrázek 5: Relace include (zdroj:[6])

Obrázek 5 zobrazuje use case Kontrola kreditní karty, který testuje platnost čísla kreditní karty a výše konta na účtu, nutná pro provedení transakce. Protože je součástí Zaplacení poplatku, je mezi nimi vztah include. [7]

- ***Extend (rozšíření)***

Tímto typem relace přidává rozšiřující případ užití nové –doplňkové chování do základního případu užití. Podstatným rozdílem oprati relaci include však je, že základní případ užití je sám o sobě zcela soběstačný. Deklaruje pouze tzv. body rozšíření (extension points), které nejsou součástí scénáře, ale pouze ukazují místo ve scénáři, kde eventuálně může být funkčnost rozšířena. Rozšiřující případ užití tedy přidává chování k základnímu případu užití právě v tomto bodě rozšíření. Jinými slovy, vztah extend modeluje volitelné části případů užití, rozšiřující případ užití ví, jak se přidat do základního scénáře, základnímu scénáři je jedno, kdo ho rozšiřuje. [4]

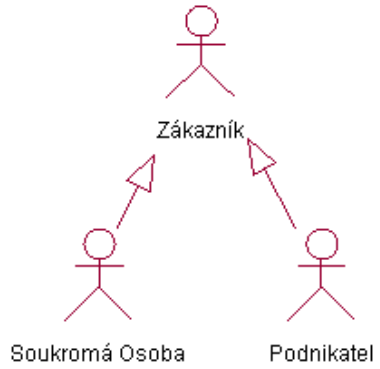


Obrázek 6: Relace extend (zdroj:[7])

Na obrázku (Obrázek 6) je zobrazen use case Zaslání upomínky čtenáři, který rozšiřuje Kontrolu výpůjček – ke spuštění dojde, pouze pokud doba, po kterou si knihy ponechává čtenář, překročí stanovenou lhůtu. [7]

- ***Generalizace (zobecnění)***

Používá se, pokud má několik aktérů nebo případů užití nějakou podobnost, např. aktéři se mohou odlišovat využitím části jiných případů užiti. [7]



Obrázek 7: Relace generalizace (zdroj:[7])

Obrázek 7 zachycuje situaci, kdy aktéři Soukromá Osoba a Podnikatel dědí případy užití od aktéra Zákazník.

2.2.3 Scénář

Případ užití je sada scénářů, které spojuje dohromady společný cíl. Scénář je sekvence kroků popisujících interakci mezi aktérem a systémem. [4] Hlavní scénář popisuje hlavní tok, je to situace, kdy vše probíhá dle očekávání. Vedlejší scénář (rozšiřující scénář) je spuštěn pouze za určité podmínky specifikované v základním scénáři, zachycuje chyby, rozvětvení nebo přerušení. Rozvětvení může snížit počet vstupních podmínek. Pokud se vyskytnou jednoduché odchylky, vytvoří se větvení v hlavním scénáři. Při výskytu složitých odchylek se vytvoří alternativní scénář. Ve scénáři se používají klíčová slova: [5]

POKUD – nová větev hlavního scénáře

JINAK – při nesplnění podmínky v předchozím klíčovém slovu

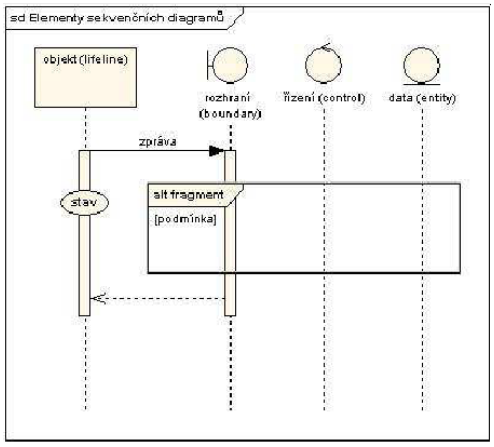
PRO .. OPAKUJ – modelování opakování

DOKUD – modelování posloupnosti na základě stanovené podmínky [5]

2.2.4 Sekvenční diagram

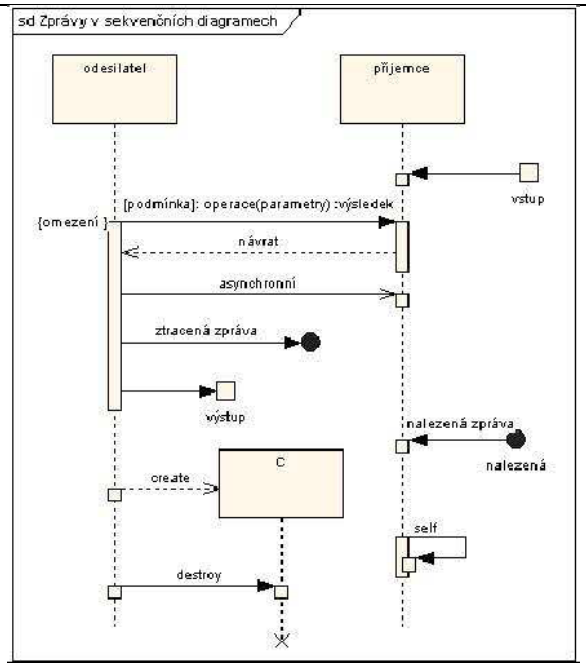
Sekvenční diagram patří do skupiny diagramů interakce. Identifikuje základní vnitřní dynamiku aplikace a potřebné metody jednotlivých tříd. Diagramy interakce popisují spolupráci skupiny objektů za účelem specifického chování. Sekvenční diagramy patří mezi nejčastěji používané diagramy interakcí. Tento typ diagramu popisuje chování objektů v rámci jednoho scénáře. Sekvenční diagramy ukazují, jak objekty komunikují navzájem mezi sebou v časové rovině. To zahrnuje rozšíření pohledu na systém o další důležitou proměnnou – čas. Interakce mezi objekty probíhá v určité posloupnosti, tato posloupnost proběhne od začátku do konce za určitý časový interval. Sekvenční diagramy nejsou vhodné pro zobrazení detailů algoritmu a precizní definici chování, ale podávají jasný obraz o činnosti několika účastníků během modelované interakce daného případu použití. Přehled základních elementů sekvenčních diagramů obsahuje Tabulka 2. [7]

Tabulka 2: Elementy sekvenčních diagramů (zdroj:[1])

Element	UML sémantika	UML syntaxe
Objekt, instance, aktér	Dokumentuje instance objektů, které participují při interakci	
Zpráva	Objekty si posílají zprávy, zpravidla jako žádosti o provedení nějaké akce	
Stav	Dokumentuje stav objektu během scénáře	
Fragment	Dokumentuje alternativní možnosti, umožňuje strukturovaní scénářů	

Zprávy, které si objekty posílají, mohou být synchronní, či asynchronní. Podrobnosti popisuje Tabulka 3.

Tabulka 3: Zprávy v sekvenčních diagramech (zdroj:[1])

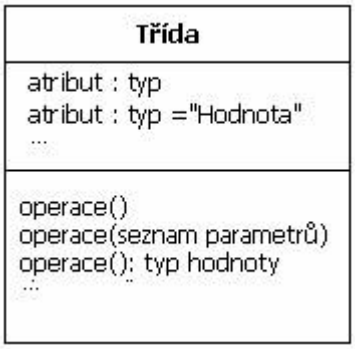
Element	UML sémantika	UML syntaxe
Synchronní zpráva	Odesilatel čeká na odpověď	
Asynchronní zpráva	Odesilatel nečeká na odpověď	
Návratová zpráva	Návrat ze synchronního volání	
Konstrukce a destrukce objektu	Odesilatel vytváří, příp. ruší objekt (destrukce)	
Nalezená zpráva	Zpráva přichází z prostředí mimo rámec této interakce	
Ztracená zpráva	Zpráva nedosáhla adresáta	
Zpráva self	Objekt zasílá zprávu sám sobě	

2.2.5 Diagram tříd

Diagram tříd obsahuje kromě tříd také různá rozhraní a seskupení. Třída (Obrázek 8) je skupina objektů, které mají podobné atributy a obecné chování. Objekt je pak instancí třídy – specifická věc se specifickými hodnotami atributů a chováním.

Oddělená část je určena pro seznam atributů třídy. Atribut je vlastnost třídy, která popisuje škálu hodnot, platnou v určitém směru pro daný objekt. Atributy začínají malým písmem. Pokud se název atributu skládá z více slov, slova se spojí do jednoho a každé slovo kromě prvního začíná velkým písmenem. U atributů lze také specifikovat jejich typ jejich hodnoty. Také lze nastavit základní hodnotu atributu. Následuje

oddělená část určená pro seznam operací, které tato třída může vykonávat. Způsob pojmenování operací je stejný jako u atributů. Dodatečné informace lze přidat zápisem parametru, se kterým bude operace pracovat, do závorek operace. Pokud je operace funkcí, musíme specifikovat typ vrácené hodnoty. [7]



Obrázek 8: Třída (zdroj:[7])

Před atributy a metodami může být také definována viditelnost. Viditelnost definuje, odkud bude atribut viditelný a přístupný. Následující tabulku (Tabulka 4) zobrazuje možné typy viditelností v class diagramu. [6]

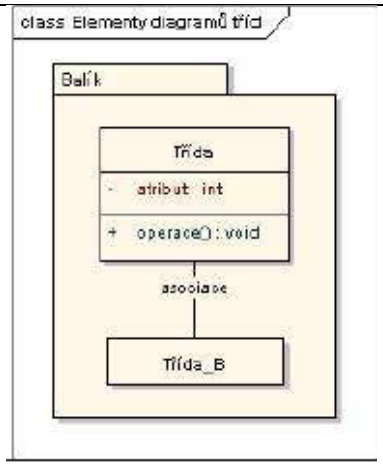
Tabulka 4: Typy viditelnosti (zdroj:[6])

Značení	Název	Popis
+	veřejný (public)	ze všech tříd
-	soukromý (private)	jen uvnitř třídy
#	chráněný (protected)	uvnitř třídy a jejich potomků
~	balíček (package)	jen z elementů uvnitř balíčku (ve kterém je tato třída) a z vnořených balíčků

Následující tabulka (Tabulka 5) zobrazuje elementy, které se vyskytují v diagramech tříd. Vztahy lze dále členit na asociaci, agregaci, kompozici a generalizaci.

Tabulka 5: Elementy diagramů tříd (zdroj:[1])

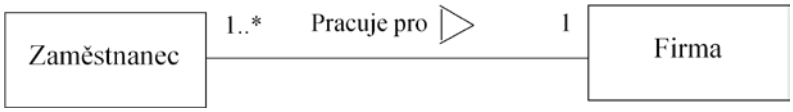
Element	UML sémantika	UML syntaxe
Třída	Dokumentuje entitu – datový typ, příp. jeho	

Element	UML sémantika	UML syntaxe
	atributy (vlastnosti) a operace (chování)	
Balík	Umožňuje vyznačit logickou skupinu tříd	
Vztahy	Dokumentuje možnost existence vztahu mezi třídami	

Vztahy

- ***Asociace***

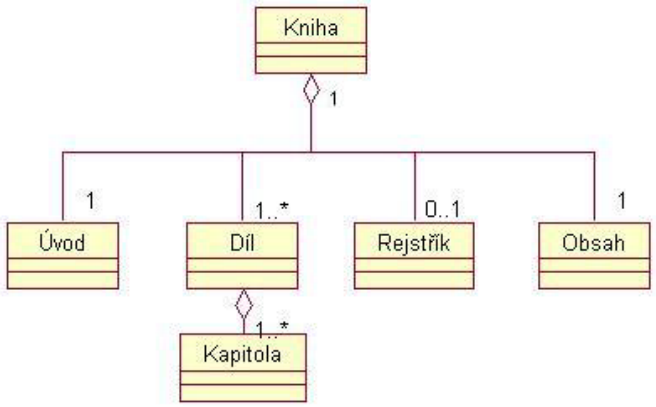
Asociace (Obrázek 9) je nejobecnější vztah mezi třídami, kdy objekty jedné třídy posílají zprávy objektům druhé třídy. V diagramu tříd se vyznačuje neorientovanou spojnicí. Kvantifikace charakterizuje různé počty instancí na obou stranách. [14]



Obrázek 9: Asociace (zdroj: [14])

- ***Agregace***

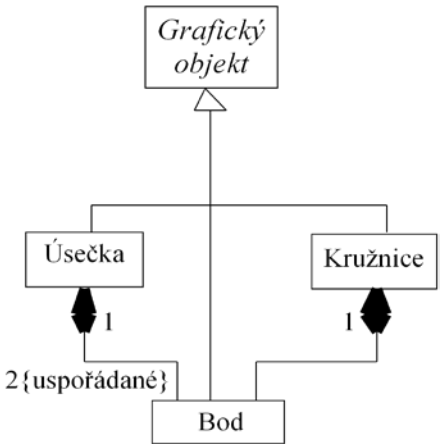
Agregace (Obrázek 10) je zvláštním druhem vztahu, kdy se vlastní třída skládá za skupiny komponentních tříd. Zjednodušený příklad: Knihu tvoří kapitoly a každá kapitola se skládá ze stránek. Agregace je reprezentovaná jako hierarchie, kde na vrcholu se nachází třída, která představuje celek, a pod ní komponenty této třídy. Na straně třídy celku je vztah ukončen kosočtvercem, od kterého vedou spojovací čáry ke komponentním třídám. [7]



Obrázek 10: Agregace (zdroj:[7])

- ***Kompozice***

Kompozice (Obrázek 11) je silnější formou agregace. Každá komponenta, tj. každá instance jedné třídy může být složkou pouze jedné instance jiné třídy. Pokud je objekt existenčně (tj. svou logikou) závislý na hodnotách složek a nemůže bez nich fungovat, jedná se o kompozici – pokud se bez nich obejde, jedná se o agregaci. Kompozici vyjadřujeme šipkou směřující od komponenty k celku a zakončenou vyplněným kosočtvercem. [14]

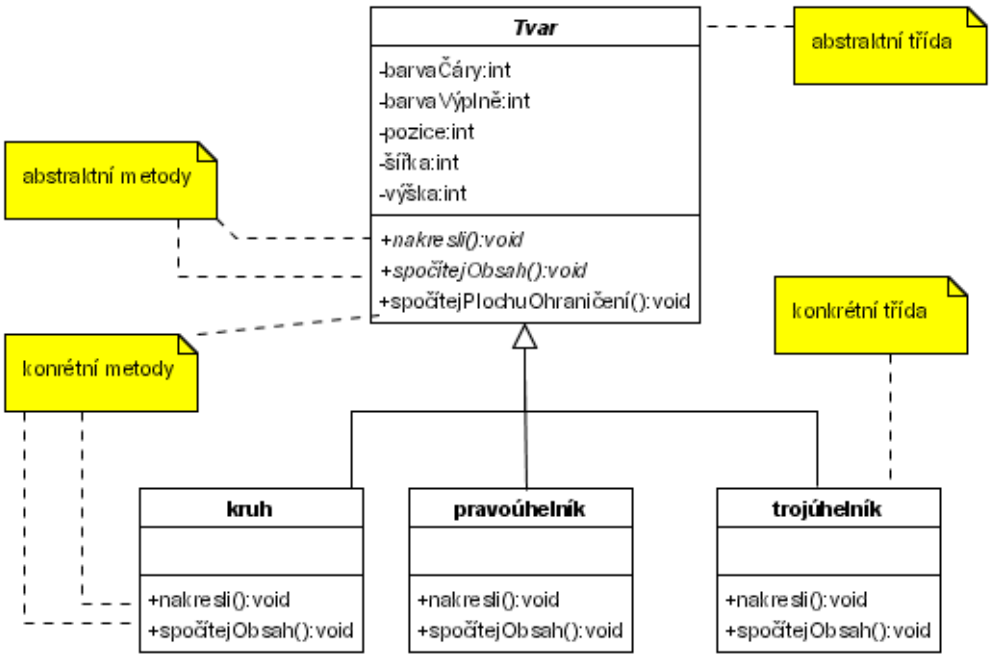


Obrázek 11: Kompozice (zdroj: [14])

- ***Generalizace***

Generalizace (Obrázek 12) se především používá z důvodů opakovaného použití, aby se nemusely opakovat společné prvky. V podstatě je generalizace vztahem mezi obecnější objektovou třídou a více upřesněnými objektovými třídami, které následují v hierarchii dědění na nižší úrovni. [4]

Abstraktní třída (Obrázek 12) je zvláštní třída, protože ve vývojovém prostředí nebude nikdy vytvořena její konkrétní instance (objekt). Tato třída obsahuje abstraktní metody, které jsou prázdné (neimplementované). Metodě, která není abstraktní, pak říkáme konkrétní metoda, třídě, která není abstraktní, pak říkáme konkrétní třída. [6]



Obrázek 12: Zobrazení abstrakce tříd a metod (zdroj:[6])

Na obrázku (Obrázek 12) je vidět další vlastnost objektového přístupu a to je polymorfismus. Polymorfní metody jsou takové metody, které se vyskytují u více tříd (tj. mají u různých tříd stejnou signaturu), ale mají odlišnou implementaci (tak jak je tomu vidět u metody nakresli). [6]

Násobnost udává, jaký počet objektů se v libovolném okamžiku účastní vztahu. Tabulka 6 zobrazuje možnosti při zobrazení násobnosti vztahů.

Tabulka 6: Násobnost vztahů (zdroj:[7])

Typ vztahu	Notace UML
Jedna ku mnoha	1:M
Jedna ku jedné nebo nula	1:1/0
Jedna ku (rozsah intervalu)	1:(2-5)
N ku M	N:M

2.3 CASE nástroje

CASE (Computer Aided Software Engineering) nástroje jsou nástroje pro podporu analýzy a návrhu aplikací. V současnosti všechny ve světě rozšířené objektově orientované CASE nástroje vycházejí z modelovacího jazyka UML. Dnes je již zřejmé, že při návrhu rozsáhlejších informačních systémů se bez použití těchto nástrojů neobejdeme, i když diagramy UML je možné zachytit i obvyklými kreslícími programy (např. VISIO). Pouze plnohodnotné CASE nástroje ale umožní propojení jednotlivých technik UML a sdílení modelů mezi členy týmu. [4]

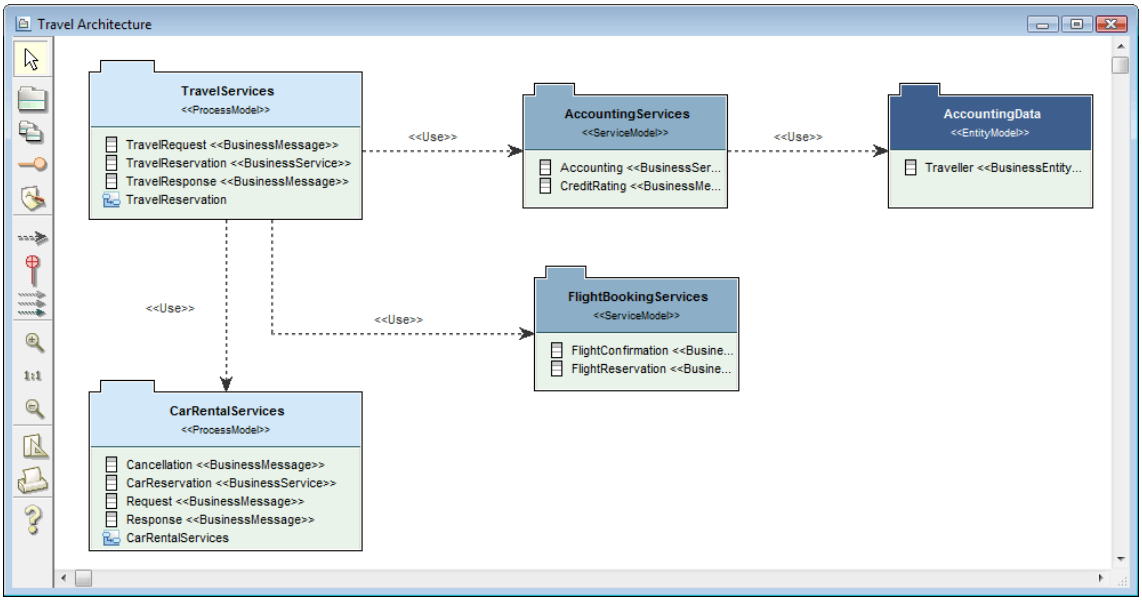
Jsou postaveny tak, aby podporovaly týmovou práci při vývoji systému, zajišťují sdílení rozpracovaných fragmentů, správu vývoje, sledují konzistenci modelu systému, automatizují některé procesy, hlídají dodržování metodiky, atd. [16]

V několika bodech lze hlavní přínosy CASE nástrojů shrnout následovně:

- Podpora tvorby katalogu uživatelských požadavků (specifikace systému);
- Podpora analytických a návrhových postupů;
- Údržba projektové dokumentace k vyvíjenému IS;
- Automatické generování programového kódu;
- Automatické generování struktury databáze v daném databázovém prostředí;
- Podpora testování systému. [3]

2.3.1 ObjectiF

Case nástroj objectiF je jedním z programů, které vyvinula německá firma microTOOL. ObjectiF umožňuje vytvářet UML modely, modelovat podnikové procesy a automaticky generovat programový kód. K vytváření programového kódu existují speciální edice objectiFu pro jazyky Visual Basic, JAVA, C# a C++. Obrázek 13 zobrazuje příklad rozhraní v programu objectiF. [12]



Obrázek 13: Program objectiF (zdroj: [12])

3 Určení postupu – sekvence kroků, volba nástrojů

Cílem bakalářské práce bylo vytvořit příklady, které budou demonstrovat objektovou analýzu pomocí case nástroje objectiF. Pro tvorbu příkladů jsem navrhl následující postup (schéma viz Obrázek 14):

- První krokem je volba skladby tématik. V tomto kroku jsem se rozhodoval jakým stylem pojmu analýzu, návrh a implementaci IS.
- Druhý krok představuje volbu diagramů z množiny možných diagramů UML.
- Třetím krokem je volba modelovaných reálií, což představovalo výběr vhodných entit z reálného světa, na kterých bych dokázal zobrazit elementy a vazby, vymezené v tabulce (Tabulka 7).
- Čtvrtý krok představuje volbu struktury řešení příkladů. V tomto kroku jsem se rozhodoval, zda příklady zobrazím jako linii na sebe navazujících diagramů nebo jako množinu diagramů, které na sebe nebudou navazovat.

Obrázek 14: Diagram postupu řešení (zdroj:vlastní)

3.1 Krok 1: Volba skladby tématik

Při volbě skladby tématik jsem se rozhodoval jakým stylem pojmu analýzu, návrh a implementaci IS. V této práci jsem se rozhodl především věnovat analýze, jelikož je program objectiF koncipován pro objektovou analýzu. V programu objectiF jsem vytvořil konceptuální model, který obsahoval UML diagramy. Pro účel tvorby příkladu relační databáze jsem class diagramy normalizoval a tím vytvořil technologický model. Technologický model jsem přenesl do programu MS Access, kde jsem ho následně implementoval.

3.2 Krok 2: Volba modelovacích nástrojů

Pro potřebu analýzy modelu objektově orientovaným způsobem jsem si zvolil tři druhy diagramů podle notace UML. Prvním z vybraných diagramů je diagram případů užití, který je takzvaným „základním kamenem“ každého modelu a od kterého se také

odvíjí struktura dalších UML diagramů. Tento diagram popisuje dynamickou část modelu systému podniku, což znamená jednotlivé akce, které by aktér rád žádal od systému. Pomocí tohoto diagramu jsem zmapoval možné aktéry (zaměstnance) a případy užití (akce usnadňující práci zaměstnancům). Do příkladů v praktické části práce jsem se rozhodl zařadit tyto elementy a vztahy use case diagramů: aktér, případ užití, komunikace, include, extend a generalizaci aktérů. Mezi elementy use case diagramu, které jsem se rozhodl nezařadit do příkladů, patří: hranice systému a generalizace případů užití. Důvod pro nevyužití hranic systému byl ten, že program objectiF nepodporuje tuto volbu. K nevyužití generalizace případů užití jsem se rozhodl z důvodu špatné aplikovatelnosti do praktické části práce a také z důvodu horší přehlednosti čtení use case diagramu.

Dalším prvkem, který jsem se rozhodl blíže specifikovat v této práci, byl scénář. Scénář slouží k bližšímu popisu jednotlivých případů užití, což je nutné pro pochopení významu případu užití, který by bez tohoto popisu ztrácel smysl pro účel další analýzy. V praktické části práce jsem se rozhodl ukázat hlavní a alternativní scénář, dále pak klíčová slova POKUD a JINAK. Ze základní struktury scénáře jsem si vybral podmínky, které jsou nutné splnit před zahájením (Preconditions) a po skončení (Postconditions) případu užití a nakonec akci (Trigger), která tento scénář spustí. Mezi prvky, které jsem se rozhodl nezařadit do praktické části, patří klíčová slova DOKUD a PRO… OPAKUJ. Pro toto jsem se rozhodl důvodu špatné aplikovatelnosti do daných příkladů.

Jako druhý diagram jsem zvolil sekvenční diagram. Tento diagram patří do skupiny dynamických diagramů, konkrétněji do skupiny diagramů interakce. Při tvorbě jednotlivých digramů jsem vycházel z již vytvořených scénářů, které charakterizovali případy užití. Mezi elementy a vztahy, které jsem se rozhodil zařadit do praktické části práce, patří: objekt, stav, zpráva self a synchronní zpráva. Elementy, které nebyly zařazeny do práce, jsou fragment, návratová, asynchronní, nalezená a ztracená zpráva, konstrukce a destrukce objektu. Fragment, asynchronní, nalezenou a ztracenou zprávu jsem se nezařadil do práce z důvodu toho, že je program objectiF nepodporuje. Konstrukce a destrukce objektu není obsažena v práci, protože by např. vytváření objektu zaměstnanec vyžadovalo zadat příliš mnoho parametrů, což by vedlo

k horší čitelnosti sekvenčního digramu. Návratovou zprávu jsem rozhodl nezařadit do elementů z důvodu nižší přehlednosti diagramů.

Poslední diagram, který bude sloužit k zobrazení modelu systému, je diagram tříd. Tento diagram patří do skupiny statických diagramů v UML notaci. Slouží, jak již název napovídá k zobrazení tříd. Tyto třídy jsem zmapoval z již vytvořených sekvenčních diagramů. Zmapované třídy však nebudou vystihovat všechny možné třídy, které mohou být v tomto modelu. Třídy budou obsahovat metody, které jsem zmapoval ze sekvenčního diagramu. Tyto metody v diagramu tříd představují zprávy v sekvenčním diagramu. Třídy obsahují atributy, které je nutné zmapovat z parametrů volaných metod a také při identifikaci objektů (např. jméno zaměstnance). Obsahová škála diagramů tříd je poněkud členitější, než u předešlých digramů a proto zde nebudou uvedeny úplně všechny elementy, které lze identifikovat v diagramech tříd. Mezi elementy a vztahy, na které jsem se rozhodl poukázat v této práci, patří třída, viditelnost, atributy, metody (parametry, návratová hodnota), abstraktní třída a metoda, konkrétní třída a metoda, přetěžování metod, asociace, agregace, generalizace a násobnost vztahů. Mezi popisovanými elementy a vztahy, avšak neuvedenými v praktické práci, patří balíček, kompozice a polymorfismus. Tabulka 7 zobrazuje souhrn všech elementů a vztahů, které jsem použil v praktické části práce.

Tabulka 7: Elementy a vztahy použité v praktické části (zdroj:vlastní)

Sledované prvky	**Elementy**	**Vztahy**	**Nepoužité elementy, vztahy**
Diagram případů užití	Aktér, případ užití	Komunikace, include, extend, generalizace aktérů	Hranice systému, generalizace případů užití
Scénář	Preconditions, postconditions, triggers, alternativní scénář klíčová slova: POKUD, JINAK	-	Klíčová slova: DOKUD, PRO…OPAKUJ
Sekvenční diagram	Objekt, stav	Synchronní zpráva, zpráva self	Fragment, asynchronní zpráva, nalezená

Sledované prvky	Elementy	Vztahy	Nepoužité elementy, vztahy
			zpráva, ztracená zpráva, konstrukce – destrukce objektu, návratová zpráva
Diagram tříd	Třída, viditelnost, atributy, metody (parametry, návratová hodnota), abstraktní třída a metoda, konkrétní třída a metoda, přetěžování metod	Asociace, agregace, generalizace, násobnost vztahů	Balíček, kompozice, polymorfismus

3.3 Krok 3: Volba modelovaných reálií

Druhý krok představoval volbu reálií, na kterých bude provedeno modelování zvolenými diagramy. Vybral jsem si modelovat pouze jednu reálii a to z důvodu zobrazení vazeb mezi jednotlivými prvky reálie (systému). Z množství reálií možných modelovat jsem si vybral podnikovou reálii, která je blíže popsána v kapitole 4.1 a 4.2. Tento podnik jsem modeloval jako systém obsahující prvky, komunikující s okolím a obsahující vstupy a výstupy.

3.4 Krok 4: Volba struktury řešení

Volba struktury řešení byl krok, ve kterém jsem se rozhodoval, jakým způsobem budu přistupovat k zobrazení návaznosti mezi jednotlivými diagramy. V předešlém kroku jsem se rozhodl modelovat podnik, který obsahuje prvky (např. sklad). Celý tento systém obsahuje velké množství aspektů, které je nutné zohlednit při modelování, a proto jsem se rozhodl vybrat jeden prvek podniku (systému), na kterém provedu modelování. Na tomto prvku podniku zobrazím propojenost vybraných diagramů s elementy a vztahy, které jsem se rozhodl sledovat (Tabulka 7).

4 Case nástroj objectiF pro objektovou analýzu – vlastní tvorba příkladů

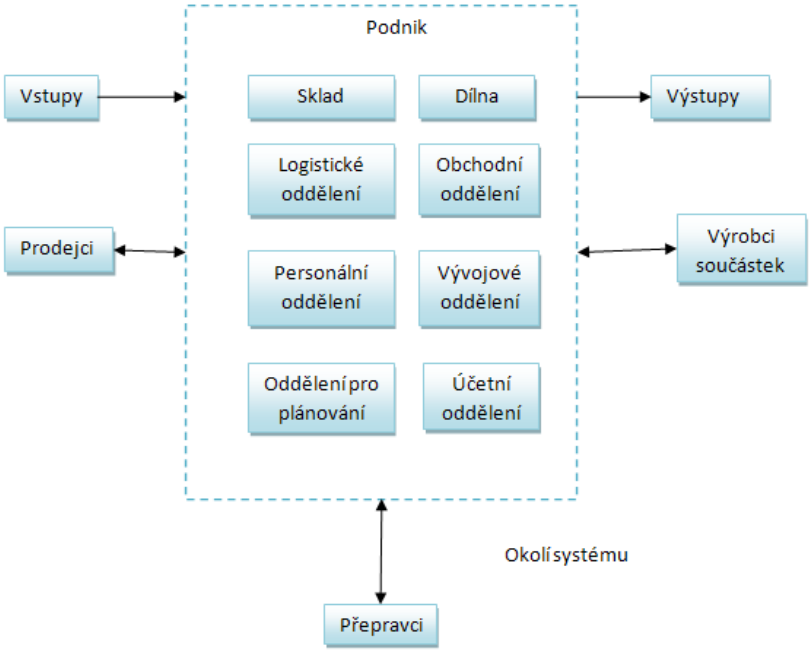
Za účelem předvedení principu objektového modelování jsem si vytvořil fiktivní firmu, na které jsem provedl analýzu. Vybral jsem si konkrétní část podniku, na které jsem provedl modelování pomocí UML diagramů.

4.1 Základní popis podniku

Firma Kola Richter s.r.o. byla založena v roce 1993, kdy se majitel Karel Richter rozhodl založit menší podnik v Jablonci nad Nisou. Výroba se převážně specializovala na montování jízdních kol. Podnik ze začátku zaměstnával minimální množství lidí nutných pro chod. Bylo zaměstnáno přibližně 50 lidí do pozic, jako jsou např. dělník, logistik, účetní. V průběhu několika let společnost dosáhla pomalého růstu, a proto si mohla dovolit modernizovat svá zařízení a také rozšířit svůj personál. Majitel se rozhodl v roce 2000 zavést do firmy IS, který by usnadnil provoz a řízení podniku. IS splňoval pouze základní funkce, ale i tak došlo k výraznému zlepšení při rozhodování a řízení firmy. S ohledem na zkušenosti s IS se pan Richter rozhodl v roce 2010 investovat do pořízení nového informačního systému, který by pokryl veškeré funkce firmy, tak aby mohl zlepšit výkon celého podniku.

4.2 Základní vymezení podnikové reálie

Podnik je členěn na několik oddělení, které spolu spolupracují a vzájemně komunikují. Struktura podniku je vidět na následujícím digramu (Obrázek 15).



Obrázek 15: Zobrazení firmy jako systému (zdroj:vlastní)

Sklad

Na tomto místě jsou uskladněny součástky nutné pro kompletaci kol v dílně a také jsou zde skladovány výrobky (jízdní kola). Skladník dodává do dílny součástky podle předem stanoveného rozpisu součástek do výroby, který plánuje vedoucí plánování. Skladník má za povinnost evidovat množství dodaných součástek do výroby a součástek dodaných od dodavatelů, a tak vést kompletní evidenci. Výrobky jsou přiváženy z dílny. Množství výrobků je dáno plánem výroby, který je vytvořen vedoucím plánování. Stejně jako u součástek má skladník za povinnost vést evidenci všech kol, které byly přivezeny z výroby.

Dílna

Dílna je místo, kde pracují dělníci, kteří kompletují jízdní kola ze součástek dodaných ze skladu. Vedoucí výroby dohlíží, aby byla dodržena denní norma výroby kol, a také dbá na celkovou plynulost provozu. O jednotlivých dělnících je vedena evidence, která pomáhá personálnímu oddělení v rozhodovacích procesech. Což představuje např. možnost prémii, nebo také nutnost k propuštění zaměstnance.

Oddělení pro logistiku

Toto oddělení se stará o optimalizaci materiálového toku ve výrobním procesu. Což představuje nutnost řídit dopravu výrobků k prodejcům, dopravu součástek od výrobců a následné uskladnění. To představuje velmi těsnou komunikaci mezi logistickým oddělením a skladem, tak aby nedocházelo k nedostatku součástek na skladu, nadbytku součástek nebo také v extrémních případech nemožnost uskladnění. Toto oddělení se stará o optimalizaci způsobu dopravy, dopravní a manipulační prostředky a stanovuje harmonogram využití těchto prostředků.

Oddělení pro plánování

Odděleni pro plánování vytváří plán výroby, rozpisy jednotlivých položek pro konkrétní oddělení. Snaží se optimalizovat a rozvrhnout náklady firmy. Stará se také o evidenci údajů souvisejících s činností oddělení.

Personální oddělení

Personální oddělení vede evidenci o všech zaměstnancích. Sleduje průběžně jejich výkony a také výkony oddělení jako celků. Vyřizuje požadavky individuálních personálních žádostí, stará se o průběžné vzdělávání zaměstnanců a jejich rekvalifikaci, vyřizuje požadavky zaměstnanců z oblasti sociální politiky. Stará se o přijímání nových zaměstnanců a také o jejich propouštění.

Účetní a obchodní oddělení

Toto odděleni se stará o ekonomickou část podniku. Provádí operace na jednotlivých účtech, zajišťuje fakturaci, kontroluje účetní doklady, zajišťuje inventarizaci majetku, má na starosti zakládání dokladů, vypočítává mzdy zaměstnancům a zakládá mzdové listy. Stará se o osobní spis zaměstnance, zajišťuje jeho zdravotní pojišťovnu a oznamuje pojišťovně nástup zaměstnance. Spravuje docházkový systém zaměstnance, jeho bonusy a změnu jeho osobních údajů. Zajišťuje agendu daní. Provádí dokladové inventury a pořizuje inventurní soupisy.

Oddělení pro vývoj kol

K tomu, aby podnik nezůstal výrobky pozadu za konkurencí, musí vyvíjet stále nové výrobky a k tomu je určeno toto oddělení, které se stará o vývoj nových kol, které by uspěly proti konkurenci. Mezi hlavní činnosti tohoto oddělení je sledování trhu s koly, testování nových prototypů kol a jejich následné uvedení do výroby.

Okolí systému

Podnik jako celek je tvořen výrobním podnikem. Okolím můžeme chápat, to co tento komplex obklopuje.

Vstupy

Jako vstupy do tohoto systému můžeme brát:

- Součástky – jsou dodávány do podniku. Následně jsou uskladněny a slouží k následné kompletaci výrobků
- Pracovní síla – je přijímána prostřednictví personálního oddělení.
- Zaplacené faktury od prodejců - jsou placeny za dodané výrobky
- Legislativa – Vztahy podniku jsou řízeny podle právních předpisů
- Nové vybavení podniku – v průběhu času je nutně aktualizovat vybavení jednotlivých oddělení

Výstupy

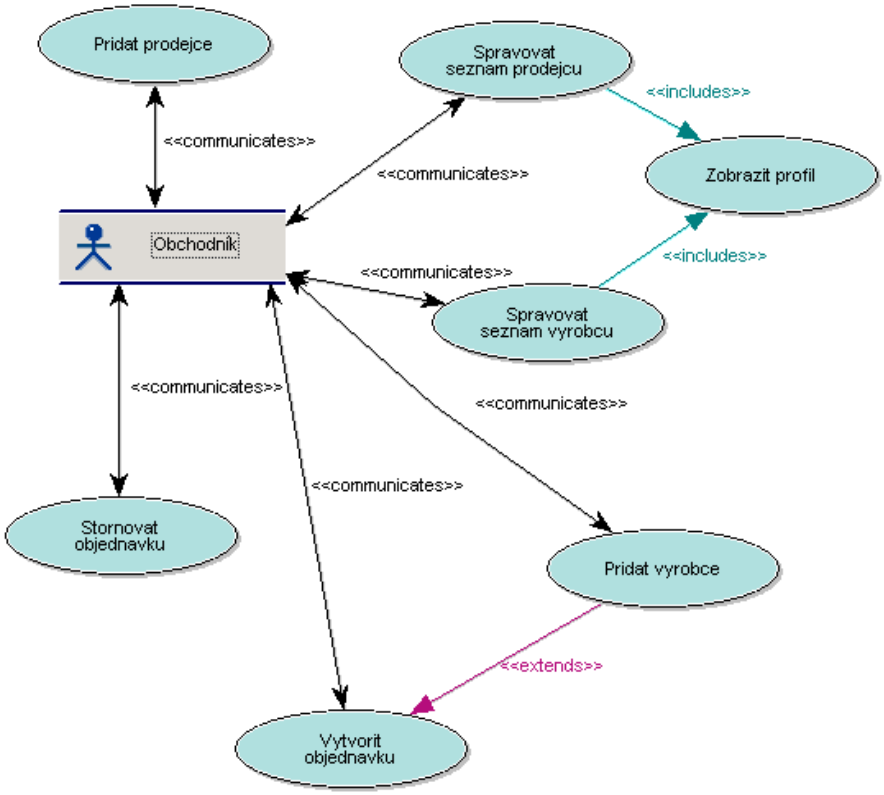
Jako výstupy do tohoto systému můžeme brát:

- Výrobky- jsou vyráběny v dílně a následně uskladněny ve skladu
- Výplaty zaměstnanců (včetně sociálního a zdravotního pojištění)
- Dodávky – jsou přepravovány přepravci
- Zaplacené faktury od podniku – jsou placeny výrobcům součástek

4.3 Provedení analýzy

4.3.1 Diagramy případů užití

Pro zobrazení sledovaných prvků jsem si vybral účetní a obchodní oddělení. V tomto oddělení je hlavním aktérem obchodník (potencionálně žádá akce od IS).

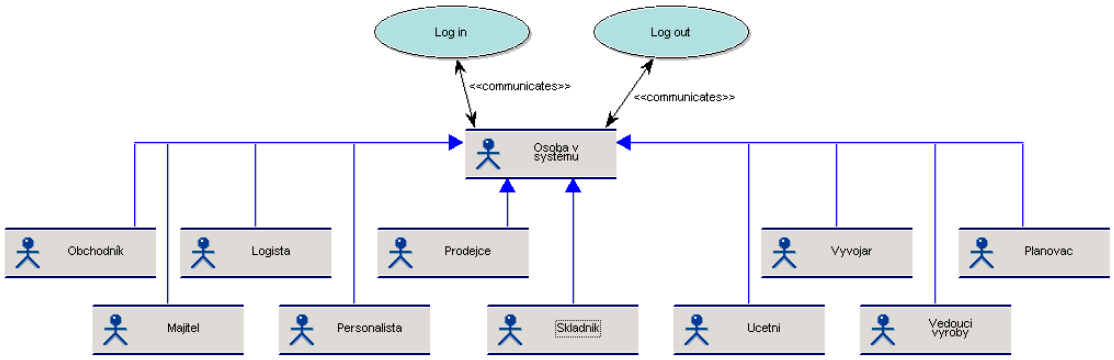


Obrázek 16: Use case diagram pro aktéra Obchodník (zdroj:vlastní)

Obchodník může spravovat seznam výrobců, od kterých firma kupuje součástky, a seznam prodejců, kteří v minulosti nakupovali od této firmy. Správa seznamu výrobců a prodejců slouží pro editaci již vytvořených záznamů. Tyto dva případy užití obsahují společnou akci – zobrazit profil. Z tohoto důvodu se tato akce vytkne do nového případu užití. Nový případ užití se následně spojí vazbou include s případy užití spravovat seznam výrobců a zákazníků. Obchodník může dále od IS žádat vytvoření nové objednávky součástek. Při této akci zadává obchodník evidenční číslo výrobce na objednávku. Pokud se jedná o nového výrobce, který není evidovaný ve firemní databázi, obchodník zvolí případ užití přidat výrobce, který přidá nového výrobce do seznamu výrobců. Případ užití přidat výrobce se spojí vazbou extend s případem užití vytvořit objednávku. Dále má obchodník možnost stornovat vytvořenou objednávku a

přidávat nové prodejce, kteří nejsou evidováni, do seznamu prodejců. Tento případ lze znázornit pomocí use case digramu (Obrázek 16).

Jednotliví zaměstnanci mají různé možnosti v tom, co mohou od IS žádat. Aby se poznalo, který uživatel žádá danou akci, je nutné, aby se uživatelé přihlašovali. Akci přihlásit se a odhlásit se mohou od IS žádat všichni uživatelé, proto je vhodné použít „reuse“ těchto akcí. K tomu je nutné vytvořit abstraktního uživatele IS, který bude žádat od IS akce přihlásit se a odhlásit se; to se nazývá generalizace aktérů (Obrázek 17).



Obrázek 17: Use case diagram zobrazující generalizaci aktérů (zdroj:vlastní)

4.3.2 Scénář

K zobrazení sledovaných elementů scénáře, jsem si vybral případ užití stornovat objednávku. Tuto akci mohou provádět obchodník a prodejce, který sice nepatří k zaměstnancům firmy, ale může se přihlásit do IS a vytvořit objednávku výrobků, které potřebuje. Jako nutná podmínka (Pre-condition) pro stornování objednávky, je existence evidované objednávky v databázi. Podmínka (Post-condition), která musí být splněna po provedení případu užití, je stornování objednávky. Spouštěč (Trigger), který vyvolá případ užití, představuje nutnost aktéra stornovat objednávku. Pro tento případ užití existují dva scénáře – hlavní a alternativní. Hlavní scénář přestavuje sekvenci kroků, která je obvyklá při spuštění případu užití. Aktér vyhledá objednávku podle ID. Může však nastat problém, že aktér nezná ID objednávky. Proto má aktér možnost zadat časový interval, ve kterém tuší, že byla objednávka vystavena. Pro tuto možnost existuje alternativní scénář. V hlavním scénáři existuje větvení, které vzniká, když se systému nepodaří nalézt objednávku podle ID. Aktér může například zadat špatně ID a tím vyvolat větvení. Použil jsem klíčová slova POKUD a JINAK. Pokud nebude nalezena objednávka podle ID, systém zobrazí prázdnou tabulku s výsledky vyhledávání.

Scénář pro případ užití Stornovat objednávku podle notace objectiF:

Short Description: Případ užití sloužící pro zrušení objednávky

Actors: Obchodník, Prodejce

Pre-Conditions: Musí být vystavena určitá objednávka

Post-Conditions: Objednávka je stornována

Trigger: Aktér chce stornovat objednávku

Scenarios

Hlavní scénář:

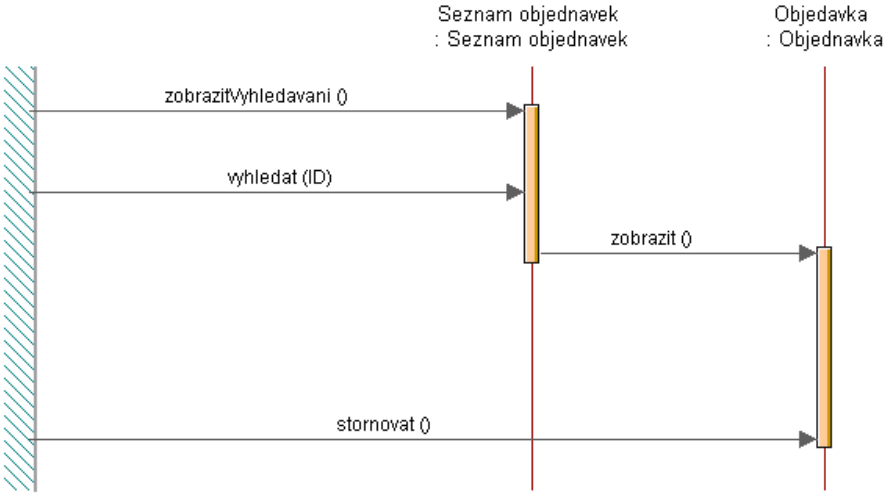
- Aktér zvolí volbu pro stornování objednávky
- Systém zobrazí dialog vyhledávání objednávek
- Aktér zadá ID objednávky do políčka pro vyhledávání
- Aktér zvolí volbu vyhledat
- Systém POKUD najde příslušnou objednávku
    - Systém zobrazí danou objednávku
    - Aktér vybere volbu stornovat objednávku
    - Systém zobrazí zprávu “Objednávka byla stornována”
- Systém JINAK zobrazí prázdnou tabulku
    - Systém se vrátí do výchozího stavu

Alternativní scénář:

- Aktér zvolí volbu pro stornování objednávky
- Systém zobrazí dialog vyhledávání objednávek
- Aktér zadá časový rozsah, kdy byla objednávka vystavena
- Aktér zvolí volbu vyhledat
- Systém zobrazí seznam objednávek
- Aktér vybere konkrétní objednávku
- Aktér vybere volbu stornovat objednávku
- Systém stornuje danou objednávku
- Systém zobrazí zprávu “Objednávka byla stornována”

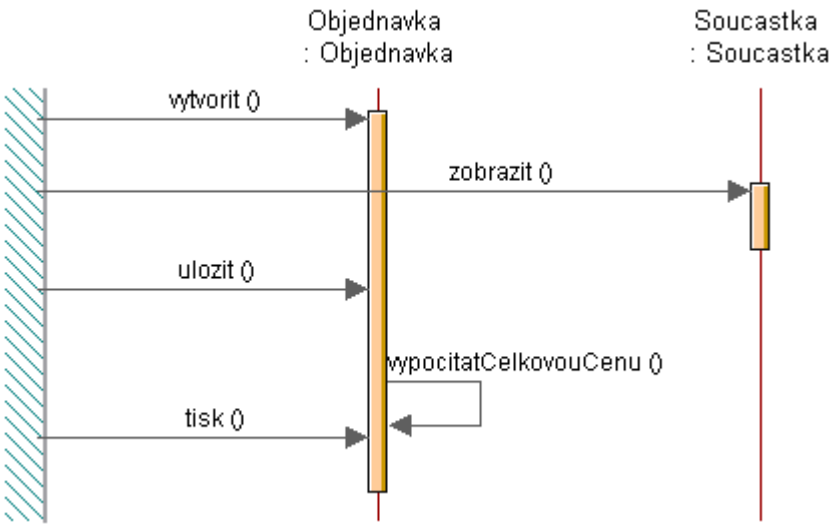
4.3.3 Sekvenční diagram

Pomocí scénáře pro případ užití Stornovat objednávku jsem vytvořil sekvenční diagram, který zobrazuje komunikaci objektů, které participují při provádění případu užití, v časovém sledu. Obrázek 18 zobrazuje hlavní scénář, který je převeden na sekvenční diagram. Každá akce, kterou provede systém je zobrazena v diagramu jako zpráva. Zprávě zobrazitVyhledavani odpovídá první akce systému. Systém zobrazí dialog pro zadávání ID objednávky. Následně aktér zadá ID objednávky a potvrdí. Zprávě vyhledat odpovídá druhá akce systému. Následně objekt Seznam objednávek pošle zprávu zobrazit objektu Objednávka. Systém zobrazí objednávku s volbou pro stornování. Aktér vybere volbu stornovat a tím pošle zprávu objektu Objednávka.



Obrázek 18: Sekvenční diagram pro stornování objednávky (zdroj:vlastní)

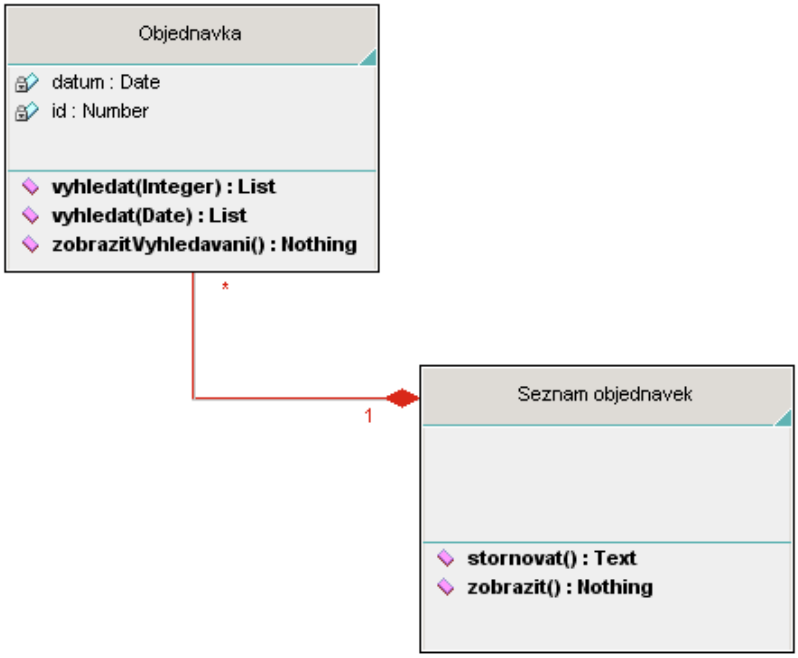
Pro doplnění všech elementů, které budu sledovat v sekvenčním diagramu, jsem vybral sekvenční diagram, který slouží obchodníkovi k vytvoření nové objednávky. Scénář pro tento případ užití má podobnou strukturu jako pro stornování objednávky, proto ho zde nebudu zobrazovat přesně, jak je uveden ve vytvořeném projektu, ale popíšu ho slovně. Na obrázku (Obrázek 19) je vidět sekvenční diagram pro Vytvoření nové objednávky. Aktér si vyžádá zobrazení dialogu pro vytvoření objednávky a seznam součástek s informacemi, které chce objednat. Po vybrání součástek zvolí aktér volbu uložit. V tuto chvíli systém uloží objednávku pomocí operace ulozit a následně se spustí self zpráva vypocitatCelkovouCenu, která vypočte cenu všech součástek uvedených na objednávce. Tato operace není volána aktérem, ale volá jí objekt, který tuto operaci vlastní. V posledním kroku si aktér vyžádá vytisknutí objednávky.



Obrázek 19: Sekvenční diagram pro vytvoření nové objednávky (zdroj:vlastní)

4.3.4 Diagram tříd

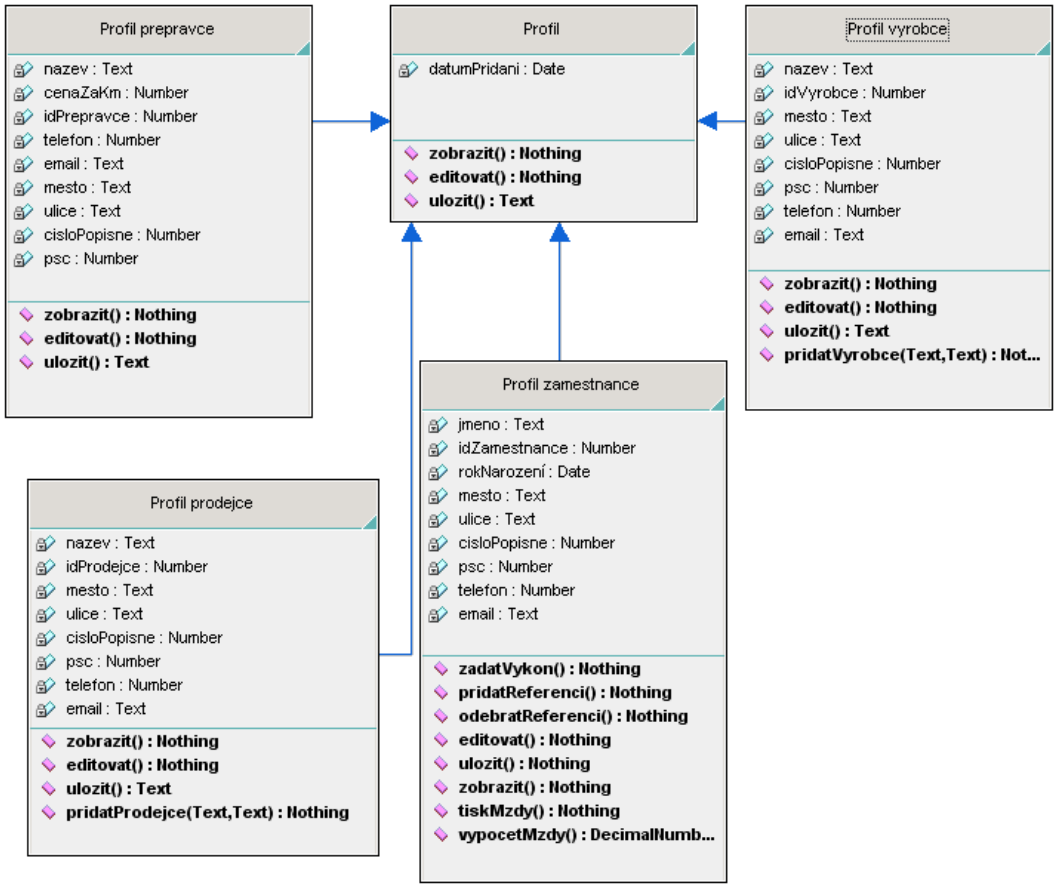
V sekvenčním diagramu (Obrázek 18) pro případ užití stornovat objednávku se vytvořili dva objekty – Objednávka a Seznam objednávek. V diagramu tříd proto vzniknou dvě stejnojmenné třídy obsahující metody, které odpovídají zprávám v sekvenčním diagramu. Na obrázku (Obrázek 20) je vidět diagram tříd, který vznikne ze sekvenčního diagramu. Třídy obsahují pouze základní metody a atributy. Viditelnost atributů je nastavena na private, což zabraňuje jiným objektům, tyto atributy přepisovat. Private atributy se zobrazí jako obdélníky se zámkem (public by se zobrazili pouze jako obdélníky). Viditelnost metod je nastavena na public, aby mohli k těmto metodám přistupovat i jiné třídy. Metoda vyhledat se vyskytuje ve třídě Objednávka dvakrát, avšak vždy s různým vstupním parametrem, který přestavuje ID objednávky v prvním případě a v druhém případě datum, do kterého se mají vyhledat všechny objednávky. Tomuto jevu se v objektově orientovaném přístupu říká přetěžování. Obě metody vrací pole hodnot. V případě ID objednávky vrací pole hodnot obsažené na objednávce. V druhém případě vrací pole možných objektů (objednávek). Třídy jsou spolu spojeny pomocí vazby agregace, kdy objednávka je částí seznamu objednávek. Dále je u této vazby zaznamenána násobnost vztahu, kdy jeden seznam objednávek obsahuje N množství objednávek.



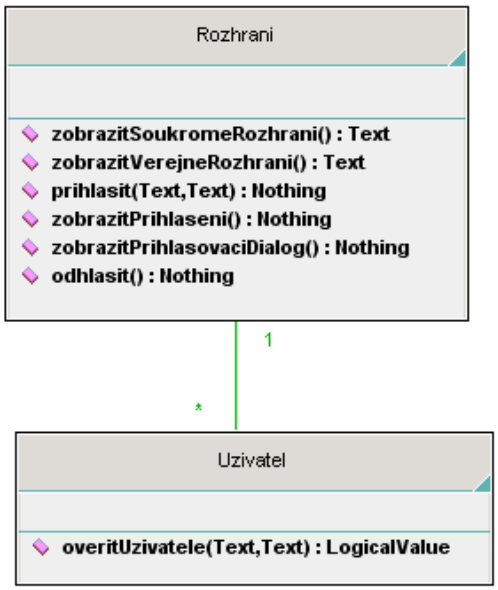
Obrázek 20: Diagram tříd vytvořený ze sekvenčního diagramu pro stornování objednávky (zdroj:vlastní)

Na následujícím obrázku (Obrázek 21) je vidět příklad abstraktní třídy Profil, která má abstraktní metody zobrazit, editovat a ulozit a atribut datumPridani. A také konkrétní třídy Profil prepravce, Profil prodejce, Profil vyrobce a Profil zamestnance. Tyto konkrétní třídy obsahují své vlastní metody a atributy i metody a atribut, které zdědily. Tyto zděděné metody, stejně s metodami vlastními se nazývají konkrétní metody (obdobně pro atributy). Konkrétní třídy jsou spojeny s abstraktní třídou pomocí generalizace.

Do IS firmy se mohou přihlásit zaměstnanci firmy a prodejci, kteří nakupují výrobky od firmy. Uživateli, který není přihlášen do IS, se zobrazí veřejné rozhraní. Uživatel může následně žádat od IS zobrazení dialogu pro přihlášení. Při odeslání loginu a hesla, dojde k ověření jeho loginu a hesla. V tomto okamžiku využívá třída Rozhraní metodu třídy Uzivatel, která ověří přihlášení. Vztah těchto tříd lze realizovat pomocí relace asociace (Obrázek 22), kdy jedna třídy využívá druhou třídu, ale není na ní existenčně závislá.



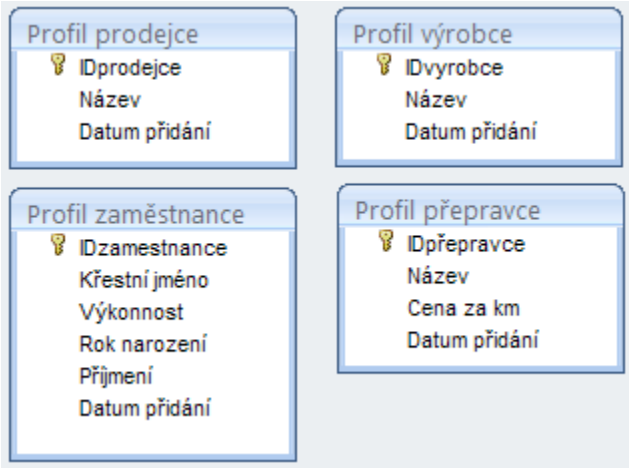
Obrázek 21: Diagram tříd zobrazující generalizaci (zdroj:vlastní)



Obrázek 22: Digram tříd zobrazující asociaci (zdroj:vlastní)

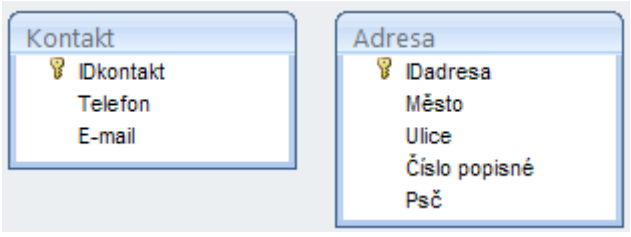
4.4 Provedení návrhu a implementace

Provedení návrhu představuje transformování konceptuálního modelu do modelu technologického. Vybral jsem si diagram tříd zobrazující generalizaci (Obrázek 21). Z tohoto diagramu vzniknou čtyři relace (Profil přepravce, Profil výrobce, Profil zaměstnance a Profil prodejce). Dále bylo nutné provést normalizaci. Nalezl jsem vícehodnotové atributy. Jméno zaměstnance lze rozdělit na křestní jméno a příjmení. Obrázek 23 zobrazuje normalizované relace pro profily prodejců, výrobců, zaměstnanců a přepravců.



Obrázek 23: Normalizované relace (zdroj: vlastní)

Vzniknou dvě nové tabulky – Kontakt a Adresa. Tabulka Kontakt obsahuje atributy telefon a e-mail. Tabulka Adresa obsahuje atributy město, ulice, číslo popisné a PSČ. V zájmu přehlednosti jsem zobrazil tabulky Kontakt a Adresa odděleně. Tyto tabulky se propojí s každou tabulkou profilů přes primární klíče, tím vznikne vazba 1:1, jelikož má každý prodejce, přepravce, výrobce a zaměstnanec pouze jednu adresu a kontakt.



Obrázek 24: Tabulky Kontakt a Adresa (zdroj: vlastní)

5 Závěr

Cílem bakalářské práce bylo vytvoření příkladů, které využijí studenti prezenčního i kombinovaného studia. Příklady jsou členěny na analýzu, návrh a implementaci v prostředí SW nástroje objectiF (demo verze).

V první části jsem charakterizoval přístupy k analýze a návrhu IS. Analýzu jsem se rozhodl provést pomocí objektově orientovaného přístupu, neboť objectiF je na tento přístup orientován. Proto jsem blíže popsal UML diagramy, které jsem použil v praktické části bakalářské práce.

Před vytvořením příkladů jsem se musel rozhodnout o volbě skladby tématik, modelovacích nástrojů, modelovaných reálií a struktuře řešení práce. Vytvořil jsem si fiktivní firmu, u které jsem popsal části, z kterých se skládá.

Při tvorbě příkladů jsem se zaměřil především na fázi datové analýzy. Na příkladech jsem zobrazil sledované elementy a vazby, které jsem si definoval. Vytvořené modely v programu objectiF obsahují use case diagramy, scénáře, sekvenční diagramy a diagramy tříd ke každému prvku firmy.

Konceptuální model vytvořený v objectiFu jsem transformoval na technologický model pro relační databázi. Při tomto kroku jsem provedl normalizaci a následně vytvořený technologický model jsem implementoval v MS Access. Ucelená linie příkladů je zobrazena v příloze této práce.

Program objectiF (demo) byl vhodný pro vytvoření dílčích praktických příkladů; byla zde však určitá omezení, neboť demo verze neobsahuje tak velkou škálu diagramů i možností tvorby elementů a vazeb v těchto diagramech. Na druhou stranu obsahuje možnost propojení s programem Visual Studio pro programování v #C nebo propojení s programem Eclipse pro programování v Javě.

6 Použitá literatura

[1] BENEŠOVSKÝ, Miroslav – RICHTA, Karel. *UML, alea iacta est!* [online]. [cit. 2011-03-25]. URL:<http://www.ksi.mff.cuni.cz/~richta/publications/TutorialUML.pdf>.

[2] JURA, Jakub. *UML*. [online]. [cit. 2011-03-25]. URL: <www.fsid.cvut.cz/cz/u12110/pis/.../OMT_UML/UML_sekvencni-2007.ppt>.

[3] KAJZAR, Dušan – POLÁŠEK, Ivan. *Tvorba informačních systémů I*. Slezská univerzita v Opavě, Filozoficko-přírodovědecká fakulta, 2003. 219s. ISBN 80-7248-214-9.

[4] KANISOVÁ, Hana – MÜLLER, Miroslav. *UML srozumitelně*. Brno, Computer Press, 2004. 176s. ISBN 80-251-1083-4.

[5] KRBÁLEK, Pavel. *OMO1*. URL: <http://www.komix.cz/cs-CZ/Tisk/Clanky/Historie/Spravny_CASE.aspx>.

[6] *OO - další vlastnosti*. [online]. [cit. 2011-03-25]. URL: <http://mpavus.wz.cz/oo/oo-trida-4.php>.

[7] PASTORČÁK, Petr. *Objektově orientované modelování systémů*. [online]. [cit. 2011-03-25]. URL: <http://orca.xf.cz/ooms/>.

[8] STEIN, René. *Návrh aplikací v jazyce UML - Unified Modeling Language.* [online]. [cit. 2011-03-25]. URL: <http://interval.cz/clanky/navrh-aplikaci-v-jazyce-uml-unified-modeling-language/>.

[9] ŠIMONOVÁ, Stanislava. *Modelování procesů a dat pro zvyšování kvality*. Univerzita Pardubice, Fakulta ekonomicko-správní, 2009. 193s. ISBN 978-80-7395-205-1.

[10] ŠIMONOVÁ, Stanislava – MYŠKOVÁ, Renáta – JIRAVA, Pavel. *Projektování informačních systémů – UML, procesní řízení*. Univerzita Pardubice, Fakulta ekonomicko-správní, 2006. 114s. ISBN 80-7194-895-0.

[11] ŠIMONOVÁ, Stanislava – PANUŠ, Jan. *Databázové systémy I*. Univerzita Pardubice, Fakulta ekonomicko-správní, 2007. 106s. ISBN 978-80-7194-988-6.

[12] *The most important features of objectiF at a glance.* [online]. [cit. 2011-03-25]. URL: < https://www.microtool.de/objectif/en/index.asp>.

[13] TIŠNOVSKÝ, Pavel. *Nástroje pro tvorbu UML diagramů*. [online]. [cit. 2011-03-25]. URL: <http://www.root.cz/clanky/nastroje-pro-tvorbu-uml-diagramu/>.

[14] *UML - Unified Modeling Languague*. [online]. [cit. 2011-03-25]. URL: <http://filip.fd.cvut.cz/download/Predn_2_YTMD.ppt?PHPSESSID=fb9b32184dd9f37ae6ca956338a149d7>.

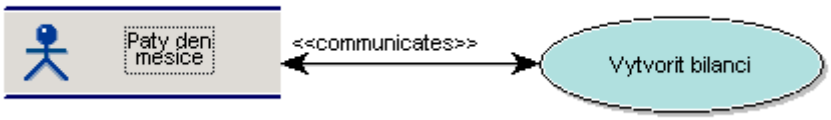
[15] *Úvod do objektového modelování a jazyka UML*. [online]. [cit. 2011-03-25]. URL: <http://www.komix.cz/Tisk/Clanky/Historie/Uvod_UML.aspx>.

[16] *Vyberte správný CASE - Stopařův průvodce CASE nástroji*. [online]. [cit. 2011-03-25]. URL: <http://www.komix.cz/cs-CZ/Tisk/Clanky/Historie/Spravny_CASE.aspx>.

Příloha A

Příklad 1 – Vytvoření bilance

Use case diagram



Obrázek 25: Use case diagram pro vytvoření bilance (zdroj: vlastni)

Scénář

Short Description

Případ užití sloužící pro výpočet bilance firmy za daný měsíc

Actors

Pátý den měsíce (počítač)

Pre-Conditions

Jsou známi hodnoty zaplacených faktur, výplat zaměstnanců, hodnota koupených součástek, částky zaplacené přepravcům a hodnota nově koupeného vybavení firmy

Post-Conditions

Je vypočítána finanční balance firmy

Trigger

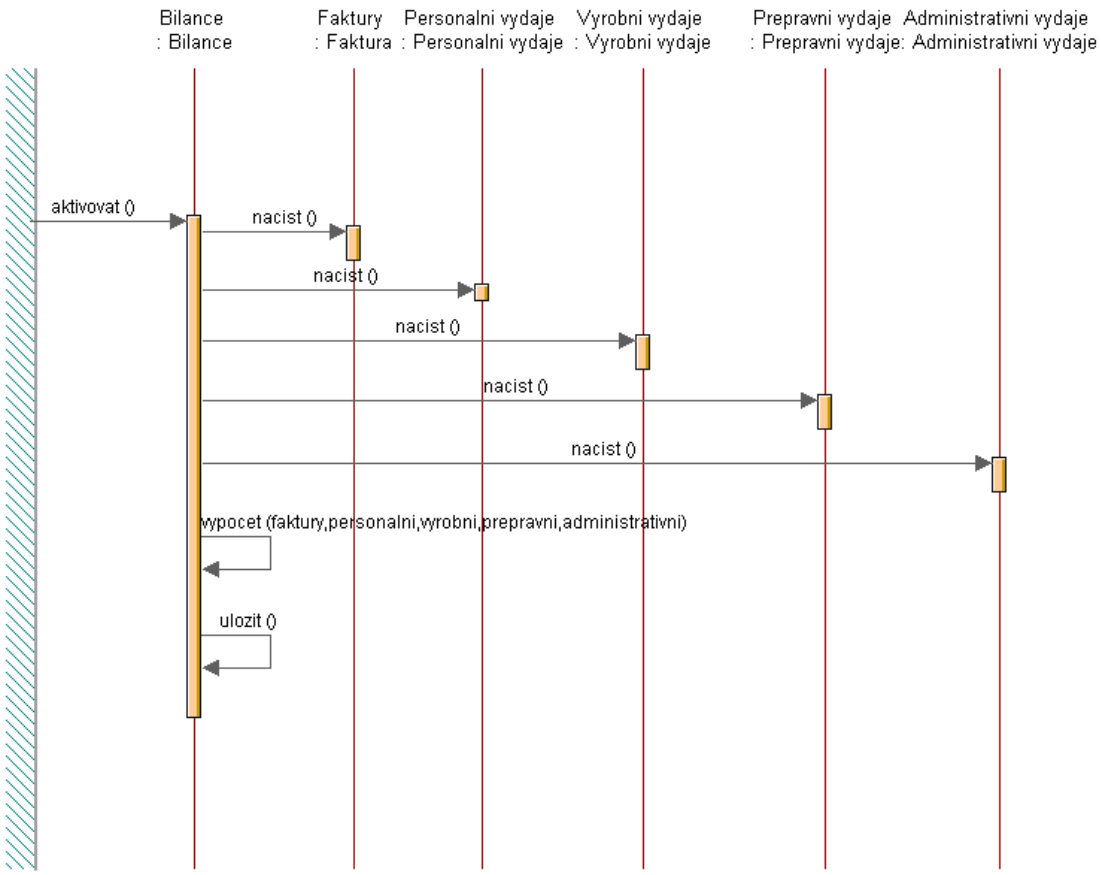
Pátý den měsíce

Scenarios

Hlavní scénář:

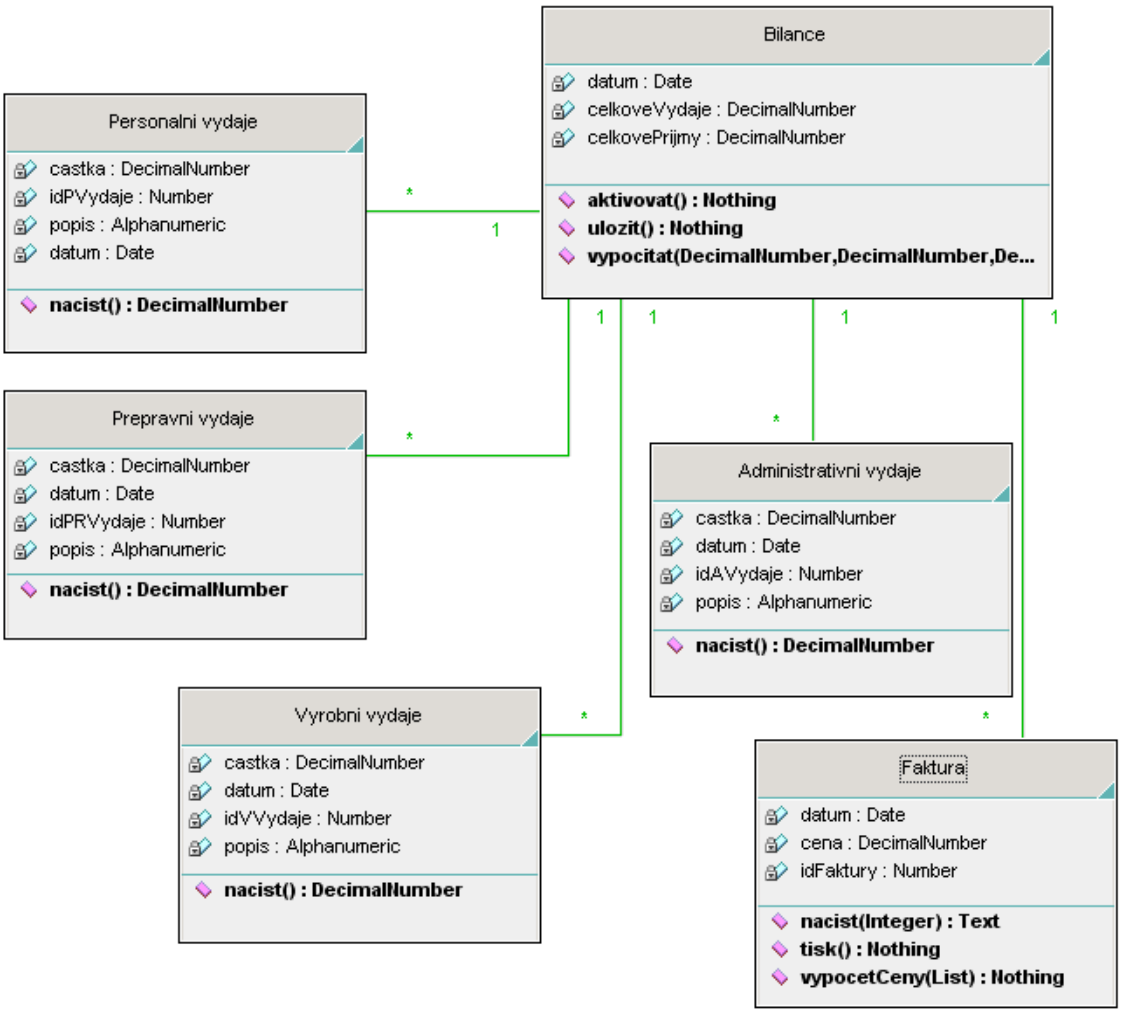
- Aktér spustí program pro vypočet bilance
- Systém načte příslušné údaje o zaplacených fakturách, výplat zaměstnanců, hodnot koupených součástek, částek zaplaných přepravcům a hodnoty nově koupeného vybavení firmy
- Systém vypočte ze zadaných údajů celkovou bilanci firmy
- Systém uloží bilanci

Sekvenční diagram



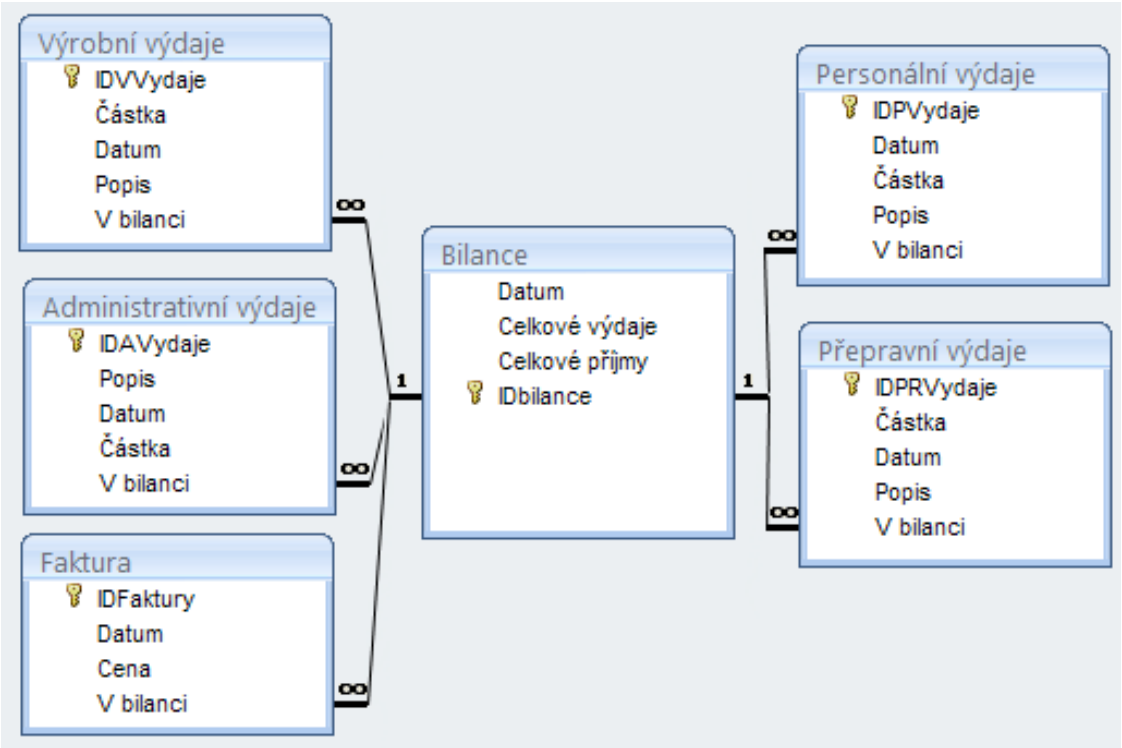
Obrázek 26: Sekvenční diagram pro vytvoření bilance (zdroj: vlastní)

Class diagram



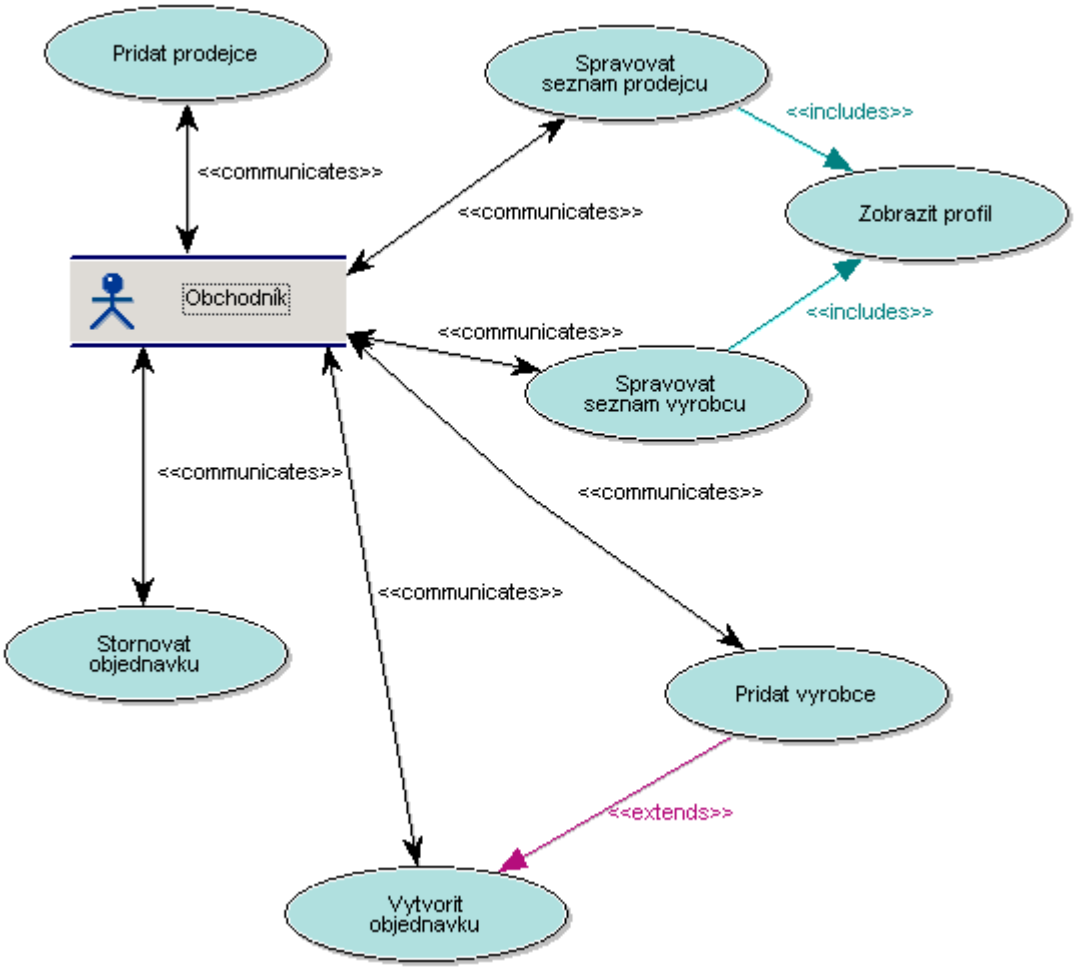
Obrázek 27: Class diagram pro vytvoření bilance (zdroj: vlastní)

Implementace v Accessu



Obrázek 28: Transformované relace z class diagramu pro vytvoření bilance (zdroj: vlastní)

Příklad 2 – Vytvoření objednávky
Use case diagram



Obrázek 29: Use case diagram pro vytvoření objednávky (zdroj: vlastní)

Scénář

Short Description

Případ užití sloužící pro vytvoření objednávky

Actors

Prodejce, Obchodník

Pre-Conditions

Aktér musí znát objekty, který chce objednat

Post-Conditions

Je vytvořena objednávka

Trigger

Aktér chce vytvořit objednávku

Scenarios

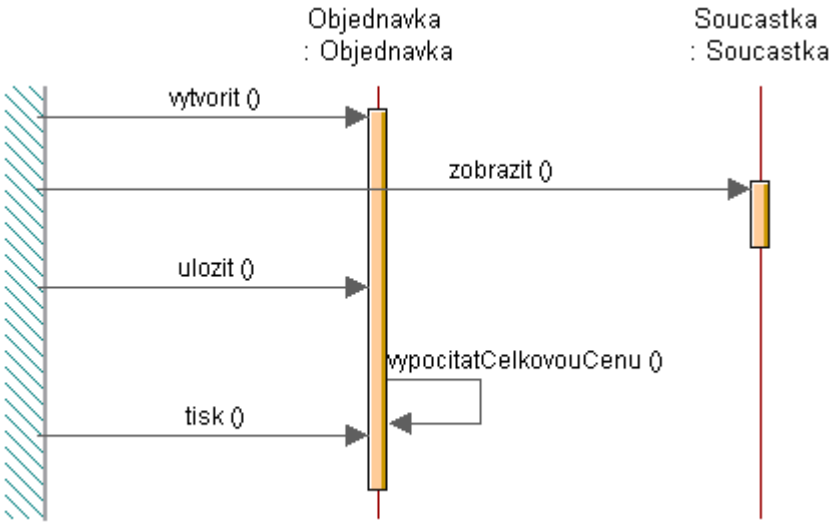
Hlavní scénář:

- Aktér zvolí volbu pro vytvoření objednávky
- Systém zobrazí dialog pro vytvoření objednávky
- Aktér vyplní zákaznické číslo, jméno, adresu a kontakt
- Aktér zvolí volbu pro zobrazení objednatelných výrobků
- Systém zobrazí tabulku položek možných objednat
- Aktér vybere položky, které chce objednat
- Aktér zvolí volbu uložit
- Systém uloží objednávku a vypočte celkovou cenu
- Aktér zvolí volbu tisk
- Systém vytiskne objednávku

Alternativní scénář:

- Aktér zvolí volbu pro vytvoření objednávky
- Systém zobrazí dialog pro vytvoření objednávky
- Aktér vyplní adresu firmy a zákaznické číslo (pokud se objednává zboží od nového výrobce – volá se use case “Pridat firmu”
- Aktér zvolí volbu pro zobrazení objednatelných položek (součástek)
- Systém zobrazí tabulku položek možných objednat
- Aktér vybere položky, které chce objednat
- Aktér zvolí volbu uložit
- Systém uloží objednávku a vypočte celkovou cenu
- Aktér zvolí volbu tisk
- Systém vytiskne objednávku

Sekvenční diagram

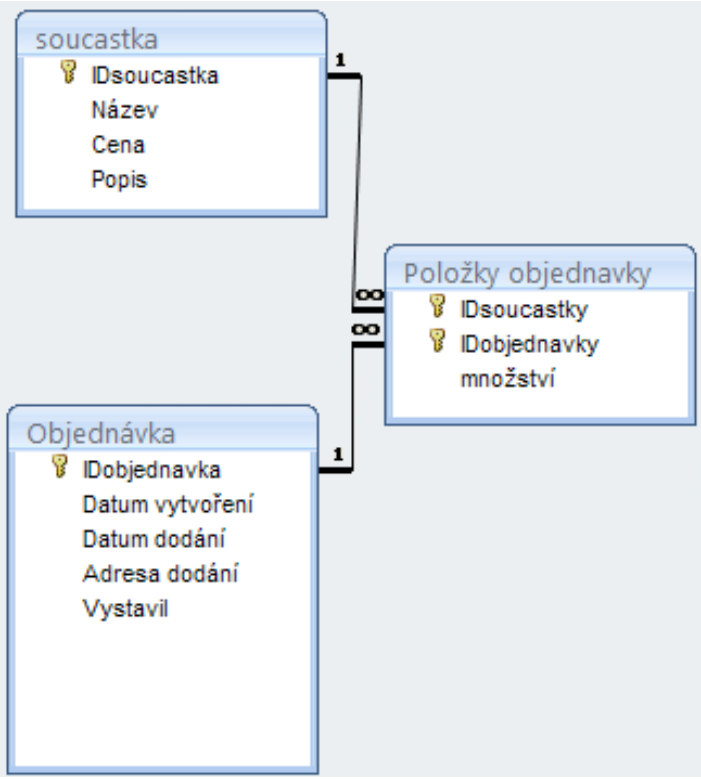


Obrázek 30: Sekvenční diagram pro vytvoření objednávky (zdroj: vlastní)

Class diagram

Obrázek 31: Class diagram pro vytvoření objednávky (zdroj: vlastní)

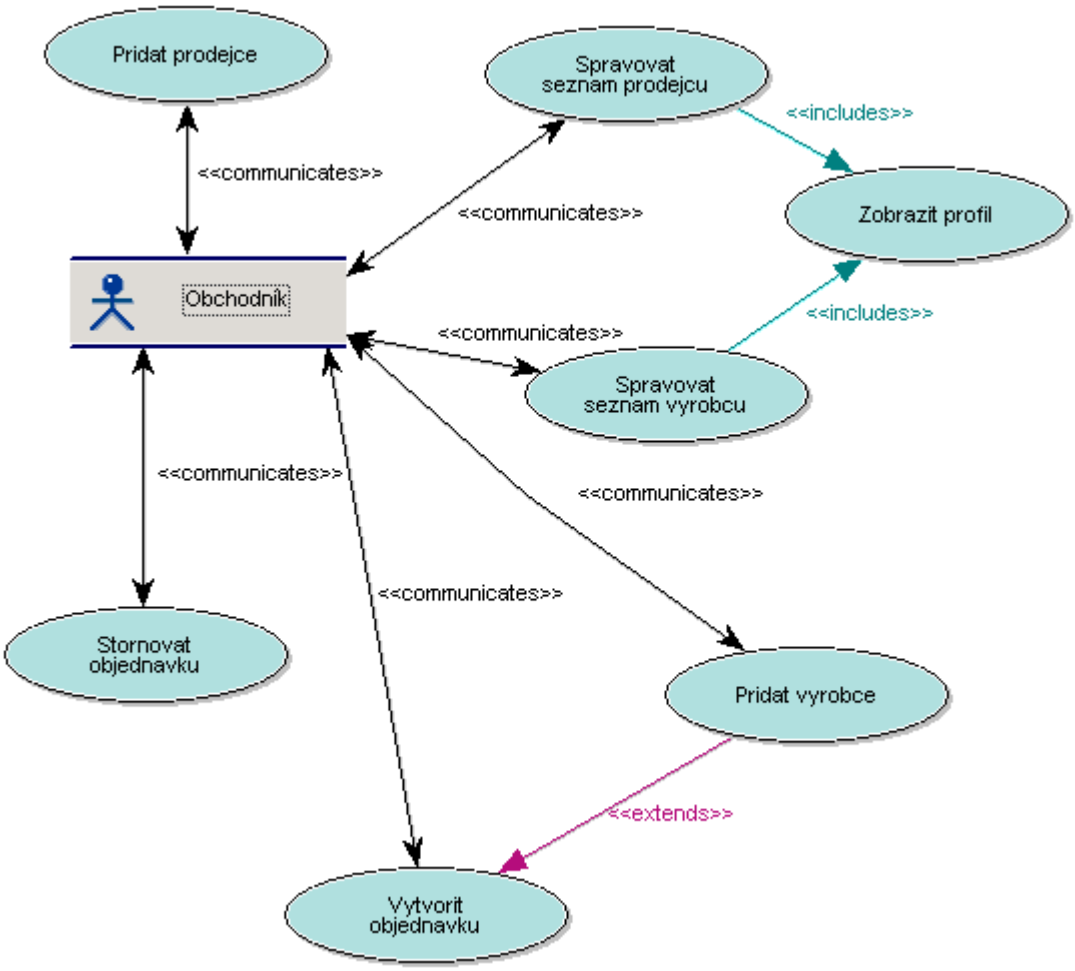
Implementace v Accessu



Obrázek 32:Transformované relace z class diagramu pro vytvoření objednávky (zdroj: vlastní)

Příklad 3 - Zobrazení profilu

Use case diagram



Obrázek 33: Use case diagram pro zobrazení profilu (zdroj: vlastní)

Scénář

Short Description

Případ užití sloužící pro zobrazení profilu zaměstnance, prodejce, přepravce a výrobce, kteří jsou evidováni ve firemní databázi.

Actors

Logista, Obchodník a Personalista

Pre-Conditions

V databázi musí být uloženy záznamy o jednotlivých osobách (zaměstnanec, prodejce, přepravce, výrobce)

Post-Conditions

Je zobrazen profil

Trigger

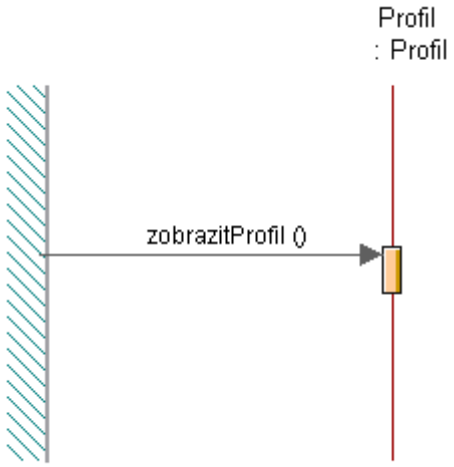
Aktér chce zobrazit profil

Scenarios

Hlavní scénář:

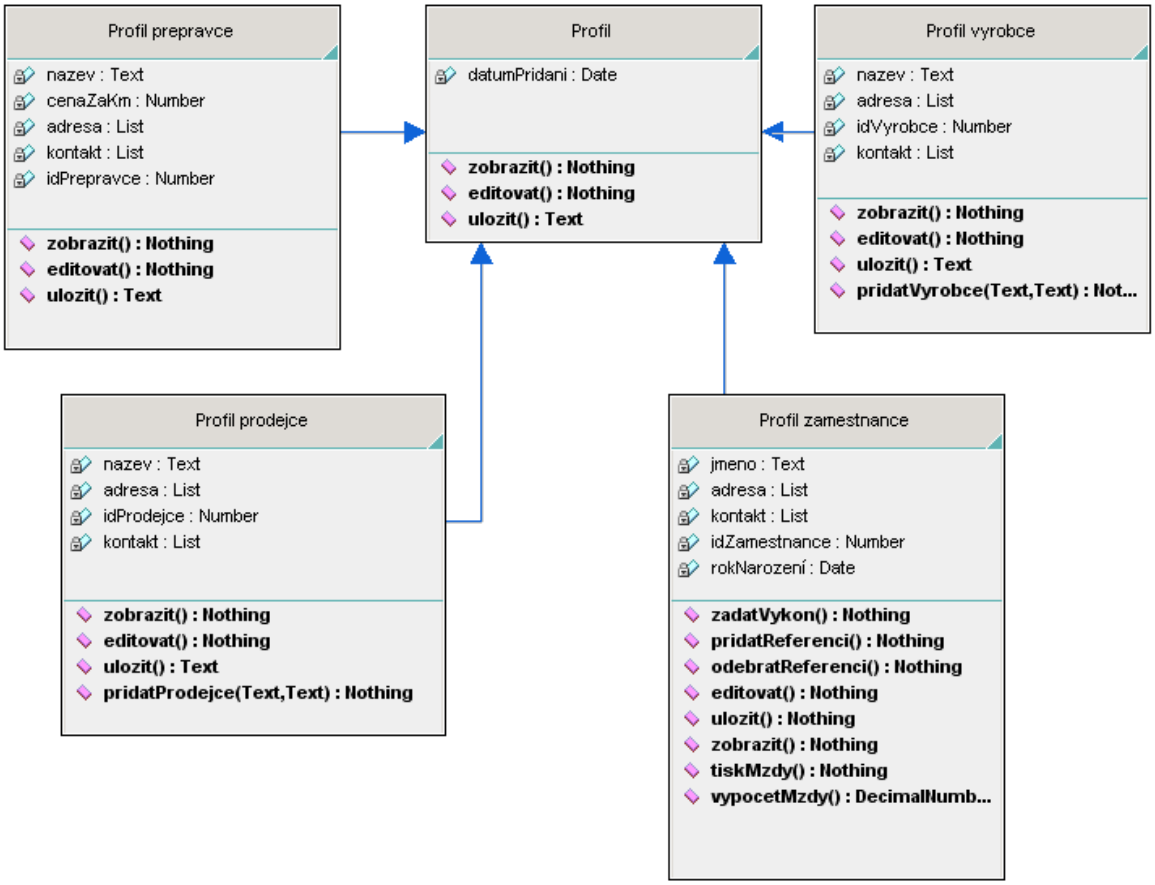
- Aktér zvolí zobrazit profil (prodejce, přepravce, výrobce nebo zaměstnance)
- Systém zobrazí veškeré evidované informace

Sekvenční diagram



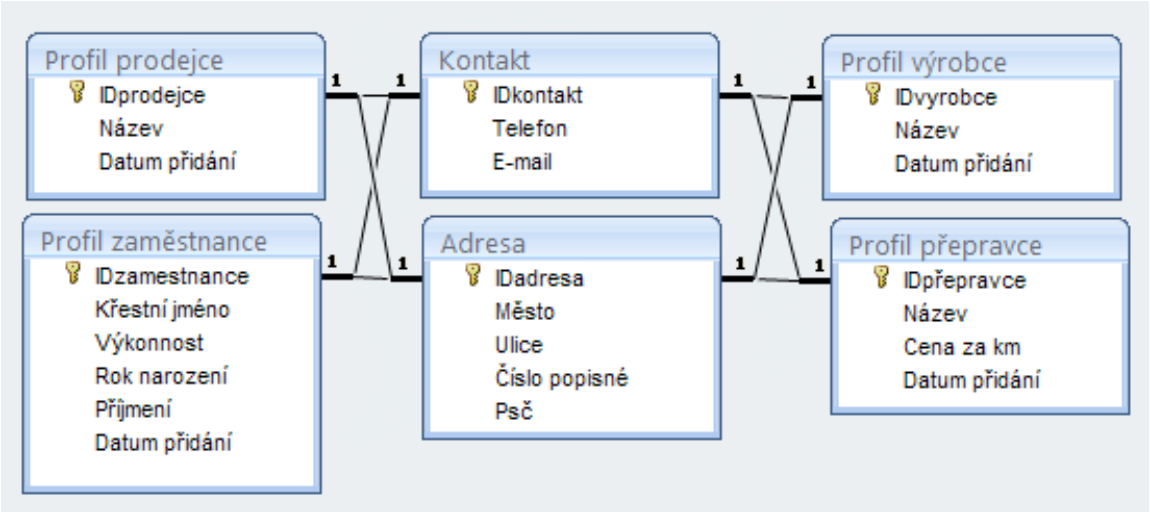
Obrázek 34: Sekvenční diagram pro zobrazení profilu (zdroj: vlastní)

Class diagram



Obrázek 35: Class diagram pro zobrazení profilu (zdroj: vlastní)

Implementace v Accessu



Obrázek 36: Transformované relace z class diagramu pro zobrazení profilu (zdroj: vlastní)

Příloha B

Obsah přiloženého CD

- BP.docx – Text bakalářské práce
- BP2.zip – Modely v programu objectiF
- Data.accdb - DB v programu Access