Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Analýza a návrh informačního systému ve vybraném podniku

Michal Voják

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačních technologií

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Voják Michal

Systémové inženýrství

Název práce

Analýza a návrh informačního systému ve vybraném podniku

Anglický název

Analysis and Design of Information System in Selected Company

---

Cíle práce

Cílem této diplomové práce je obecný popis problematiky informačních systémů, především jeho analýzy při zavádění systému do podniku.
Dalším cílem je tvorba dokumentu obsahující soupis požadavků na systém a analýzu informačního systému, která je podkladem pro společnost, zabývající se právě vytvořením nového informačního systému pro vybraný podnik.

Metodika

V první části jsou objasněny základní teoretické poznatky z oblasti informačních systémů a jeho analýz. V druhé části je proveden sběr požadavků na systém a jeho analýza. Analýza bude vytvořena za pomocí programovacího jazyka UML dle metodiky Unified Process.

Harmonogram zpracování

Teoretická část: 5. 2011 - 12. 2011
Praktická část: 8. 2011 - 2. 2012

Rozsah textové části

60 - 80 stran

Klíčová slova

Informační systém, analýza IS, sběr požadavků, UML

Doporučené zdroje informací

SODOMKA, Petr; KLČOVÁ, Hana. Informační systémy : v podnikové praxi. 2. vydání. Brno : Computer Press, a.s., 2010. 504 s. ISBN 978-80-251-2878-7.

BASL, Josef; BLAŽÍČEK, Roman. Podnikové informační systémy : Podnik v informační společnosti. 2. vydání. Praha : Grada, 2008. 288 s. ISBN 978-80-247-2279-5.

FOWLER, Martin. Destilované UML. 3. vydání. Praha : Grada, 2009. 176 s. ISBN 978-80-247-2062-3.

NEUSTADT, Ila; ARLOW, Jim. UML 2 : a unifikovaný proces vývoje aplikací. Brno : COMPUTER PRESS, 2007. 568 s. ISBN 978-80-251-1503-9.

Vedoucí práce

Šilerová Edita, Ing., Ph.D.

Termín odevzdání

březen 2012

doc. Ing. Zdeněk Havlíček, CSc.
Vedoucí katedry

prof. Ing. Jan Hron, DrSc., dr.h.c.
Děkan fakulty

V Praze dne 27.3.2012

Čestné prohlášení

Prohlašuji, že svou diplomovou práci " Analýza a návrh informačního systému ve vybraném podniku" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 4.4.2012 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Edita Šilerová, Ph.D. za odbornou pomoc při psaní této diplomové práce. Dále bych rád poděkoval celé své rodině za podporu při studiu.

Analýza a návrh informačního systému ve vybraném podniku

Analysis and design of information systems in selected company

Souhrn

Diplomová práce je zaměřena na analýzu informačního systému a návrh dalšího vývoje do dokončení analýzy. Cílem práce je popsat sběr požadavků na nově vznikající informační systém ve vybraném podniku a vytvoření analýzy z těchto požadavků dle metodiky Unifield process a nakonec vytvořit návrh dalšího postupu.

První, praktická část je zaměřena na vysvětlení základních pojmů informačních systémů, jejich životních cyklů a metodik vývoje, především metodiky Unifield Process, která slouží pro tvorbu nových informačních systémů a popisuje jakým způsobem provádět sběr požadavků a jejich analýzu s pomocí jazyka UML.

V druhé, praktické části je vytvořen dokument obsahující všechny požadavky na informační systém zapsané pomocí případů užití a podrobné analýzy těchto požadavků. Analýza obsahuje především analytické třídy a diagramy interakcí, díky kterým je popsána celková funkčnost celého, nově vznikajícího, informačního systému. Společnost díky tomuto dokumentu bude mít do budoucna jasnou představu, jak systém, který denně používají, funguje..

Summary

This thesis is focused on analyzing the information system and system and for further development to complete the analysis. The aim is to describe the collection of requirements on new information systems in selected company and the creation document of analysis these requirements according to the methodology Unifield process and create a proposal for further action.

First, the practical part is focused on explaining the basic concepts of information systems, their lifecycles and development methodologies, particularly Unifield Process methodology, which is used for creating new information systems and describes how to perform requirements gathering and analysis with UML.

In the second, practical part of the document is created containing all information system requirements using use cases and detailed analysis of these requirements. The analysis consists mainly of analytical grade and interaction diagrams, which is described by the overall functionality of the emerging information system. The result of this document in the future will have a clear idea of function how they systems, which every day use, work.

Klíčová slova: Analýza informačního systému, UML, Unified process, životní cyklus informačního systém, sběr požadavků, informační systém, metody tvorby informačních systémů, nástroje CASE, analytické třídy, případy užití.

Keywords: Analysis of Information System, UML, Unified Process, The life cycle of an information system, collection requirements, information system, Information Systems development methodology, CASE tools, analytic classes, use cases.

Obsah

1. Úvod

V dnešní době se každá střední a velká firma neobejde bez informačního systému. Době, kdy stačilo podniku vlastnit papír a tužku a s pomocí těchto nástrojů řídit celou firmu, je již nenávratně pryč.

To, jakým způsobem informační systém v podniku funguje, určuje, jak efektivně dokáže celá organizace pracovat. Pokud nebude systém pomáhat pracovníkům v jejich každodenní práci, může v podniku nastat velký problém.

Pokud se jedná o složitější informační systém, není možné jej jednoduše někde koupit. V tuto chvíli nastupují softwarové společnosti, které slibují vytvoření levného řešení na míru za minimum času. Zde však vzniká častý problém. U těchto systémů často chybí základní tvůrčí kámen a tím je správně vytvořená analýza informačního systému ušitá na míru společnosti.

Správně navržený informační systém je alfou i omegou každého většího podniku. Žádný nebo špatně navržený IS zpomaluje průběh práce v podniku, a tak nepřímo zvyšuje náklady.

Při tvorbě informačního systému na míru je velice důležité vědět, jakým způsobem správně postupovat. V posledních 40 letech proto vznikalo spousty metodik, které se věnovaly této problematice, a díky různorodosti požadavků na systémy vzniklo mnoho metodik. To zapříčinilo vzniku nové disciplíny, která řeší jakou metodiku zvolit na daný projekt, aby byl co nejúspěšnější.

Správná analýza informačního systému se dá označit jako jeden z nejdůležitějších dokumentů, který pokud je špatně zpracován, může ohrozit celou tvorbu a následující chod systému.

V této práci je vytvořena analýza informačního systému pro podnik Blue Orange a.s. Analýza je vytvořena v jazyce UML za použití nástroj CASE s názvem Enterprise Architect od společnosti Sparx systems.

2. Cíl práce a metodika

Cílem diplomové práce je vytvoření analýzy a návrhu postupu pro systém, který se stará o kompletní správu zákazníků v komplexu, zahrnující restauraci, wellnes centrum a hotel. Vedlejší cíl diplomové práce je obecný popis problematiky tvorby informačních systémů, především sběr požadavků při tvorbě nového informačního systému a následně analýza těchto požadavků na informační systém.

Teoretická část je zaměřena na definici základních pojmů informačních systémů, jejich životní cykly a metodiky vývoje. Dále bude vysvětlena metodika Unifield Process, která slouží pro tvorbu nových informačních systémů a popisuje jakým způsobem provádět sběr požadavků a jejich analýzu za pomoci jazyka UML.

Na základě praktických a teoretických poznatků je pak vytvořen dokument obsahující všechny požadavky na informační systém, zapsané pomocí případů užití a podrobné analýzy těchto požadavků. Tento dokument bude podkladem pro tvorbu nového informačního systému, který po jeho dokončení vystřídá starý, již nevyhovující systém společnosti.

V době zahájení diplomové práce probíhá sběr požadavků na systém a tvorba analýzy, která je součástí diplomové práce. Pro sběr požadavků jsou použity metody konzultace a zkoumání stávajícího systému. Celý projekt je vytvářen za pomocí metodiky Unifield Process v jazyce UML.

Výstupy pro tuto analýzu budou tvořeny v softwarovém nástroji CASE od společnosti Sparx systems s názvem Enterprise Architect.

3. Přehled řešené problematiky

V této kapitole je definován popis základních teoretických poznatků z oboru informačních systémů (dále jen IS) a jejich tvorby.

3.1. Co je to informační systém

Informační systém je konzistentní uspořádaná množina komponent, které spolu spolupracující za účelem tvorby, shromaždování, zpracování a rozšiřování informací. Prvky informačního systému jsou lidé a informatické zdroje. Komponenta je tvořena jedním nebo více prvky. (Ratzan, 2004).

Buchalcevová ve své knize uvádí konkrétnější definici.

„*Informační systém je systém, jehož prvky jsou informační a komunikační technologie, data a lidé.*

Cílem informačního systému je efektivní podpora informačních a rozhodovacích procesů na všech úrovních řízení organizace“. (Buchalcevová, 2005)

3.1.1. **Data, informace, znalost**

Jak popisuje Gála a spol. ve své knize, data prezentují vlastnosti objektů. Data jsou množina, které popisují objekt bez jakéhokoliv kontextu. (2009)

A dále uvádí, jakým způsobem se data transformují do informace a následně informace na znalost:

„Data se stávají informacemi, když je vhodně zpracujeme (strukturujeme) a dodáme za určitým účelem. Data v kontextu jsou informacemi a informace, které jsou použity, jsou znalostí, tzn., že zkušenosti transformují informace do znalostí.“ (Gála a spol. 2009)

3.1.2. **Proces**

„Proces je definovaný jako soubor vzájemně souvisejících nebo vzájemně působících činností, které přeměňuje vstupy na výstupy.“ (Gála a spol. 2009)

3.1.3. **Software**

Software neboli programové vybavení počítače umožňuje vytvářet na počítači užitečné činnosti, jako například kreslit, psát, sledovat video a další. (Kolář, 2005)

3.1.4. **Systém**

Pro lepší pochopení slova systém v problematice informační a telekomunikací technologie (dále jen ICT) poskytuje následující definice:

„Konzistentní a koordinovaný soubor složek vystupujících společně jako jeden celek sloužící pro společnou funkci nebo účelu. Nekonzistentní systém funguje sám proti sobě.“ (Ratzan, 2004).

Druhů informačních systému je celá řada a vždy se dělí podle různých kritérií. Tato práce je zaměřena na analýzu informačních systémů především v podnikové praxi, následující kapitola se tedy zabývá především podnikovými IS.

3.2. **Podnikový IS**

„Podnikový informační systém vytváří lidé, kteří prostřednictvím dostupných technologických prostředků a stanovené metodiky zpracovávají podnikové data a vytvářejí z nich informační a znalostní bázi organizace sloužící k řízení podnikových procesů, manažerskému rozhodování a správě agendy.“ (Sodomka a Klíčová, 2010)

3.2.1. **Druhy podnikových IS**

Podnikové IS je vhodné klasifikovat dle jejich praktického uplatnění a ve shodě s požadavky na řízení procesů. Rozhodující pro klasifikaci IS je tzv. holisticko-procesní pohled.

- **ERP** (Enterprise Resource Planning) systém, zaměřený na řízení interních podnikových procesů,
- **CRM** (Customer Relationship Management) systém, starající se o procesy směřované k zákazníkovi,
- **SCM** (Supply Chain Management) systém, řídící dodavatelský řetězec, jehož součástí je systém APS, což je systém pro pokročilé plánování a rozvržení výroby,
- **MIS** (Management Information System) manažerský IS, který slouží pro sběr dat z EPR, CRM a SCM systémů a na základě jejich rozboru poskytuje informace pro rozhodovací proces podnikového managementu.(Sodomka a Klíčová, 2010).

3.3. Model životního cyklu tvorby IS

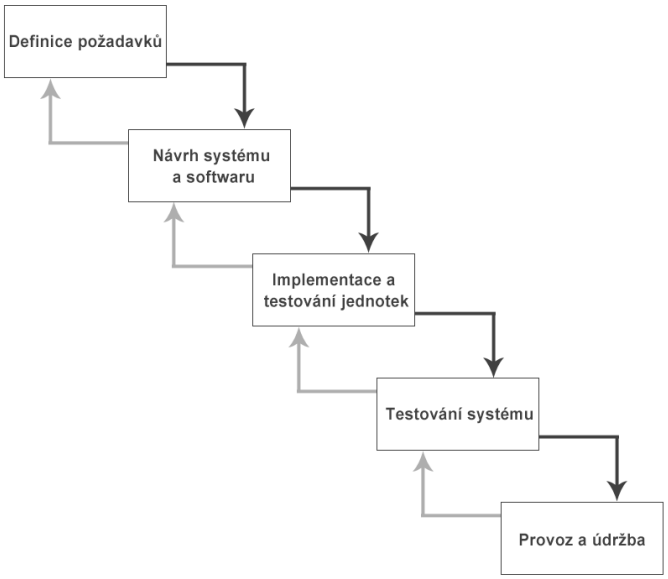
3.3.1. Sekvenční model životního cyklu

Sekvenční model je založen na myšlence, že je možné postupně projít od první fáze zadání, až do poslední fáze dokončení a předání. Nejznámější varianta sekvenčního modelu je vodopádový model. (Merunka, 2008)

Vodopádový cyklus vývoje

Jde o nejstarší přístup tvorby IS. Jeho nespornou výhodou je možnost přesného sledování postupu tvorby IS. Naopak nevýhoda spočívá s přístupem objektově orientovaného programování (dále jen OOP[1]). (Merunka, 2008)

Jak uvádí autoři Liu a Roussev ve své knize, vodopádový model života je nejrozšířenější a nejpopulárnější model a je využit v mnoha metodách tvorby IS.(2006)



Obrázek 3.1 Vodopádový cyklus vývoje (zdroj: http://www.fi.muni.cz)

3.3.2. Iterativní model životního cyklu

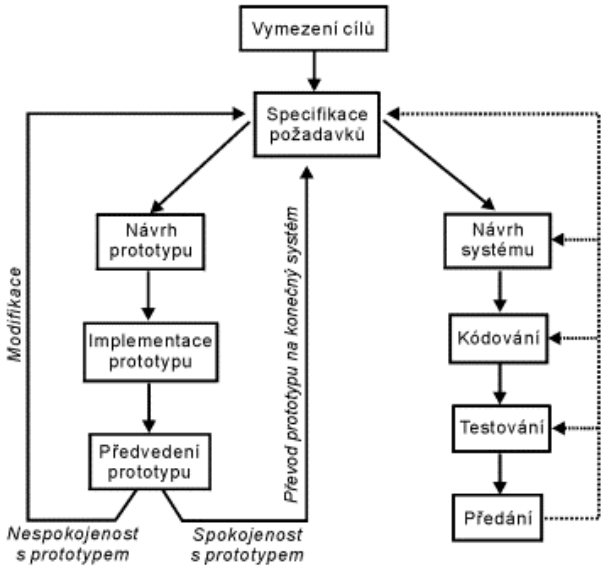
Tento model je založen na možném vracení se do předchozích fází vývoje. To je především kvůli zpřesnění zadání z minulé fáze. Mezi nejznámější modely patří prototypový a spirálový.

[1] Metodika programovaní, při které se nepoužívá klasický strukturovaný způsob, ale nahlíží se na celý problém jako na spojené objekty.

Skutečnost, že je možné začít s tvorbou dříve, než jsou upřesněné a vydefinované veškeré požadavky a využít tím zkušenosti z první implementace pro zpřesnění zadání, měla za následek velkou oblibu v komunitě, která využívá OOP. Výhoda této metody spočívá v možnosti průběžného doplňování a zpřesňování zadání. Nevýhodou je, že průběh projektu je hůře sledovatelný. (Merunka, 2008)

Prototypový přístup

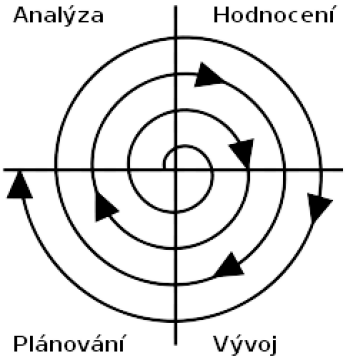
Základní charakteristikou tohoto modelu je předpoklad, že se požadavky v průběhu tvorby budou měnit. Tento model tedy umožňuje rychlou interakci na změny v zadání. Prototyp je chápán jako zjednodušená implementace systému. Tato implementace je provedena v co nejkratším čase a v takové funkčnosti, že je zákazník schopen veškeré skutečnosti snáz pochopit a reagovat tak na výsledky. Na základě připomínek je prototyp modifikován do té doby, než je zákazník s výsledkem úplně spokojen. Po této fázi nastupuje návrh a implementace systému.(Životní cyklus informačního systému, 2005)



Obrázek 3.2 Prototypový model (zdroj: http://www.fi.muni.cz)

3.3.3. **Spirálový přístup**

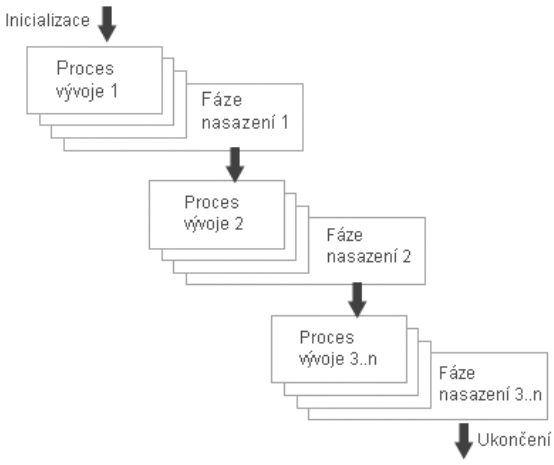
Jeho jednotlivé vývojové etapy se opakují vždy na vyšším stupni hotové problematiky. Svou strukturou může připomínat iterativní nebo inkrementální model životního cyklu. Ve spirálovém přístupu je vekou výhodou, že uživatel nedostává vždy omezenou funkcionalitu, ale pouze prototyp. Tím, že se všechny fáze neustále opakují až po samotné dokončení, objevují se základní chyby hned na začátku.(Křena a Kočí, 2006)



Obrázek 3.3 Spirálový model (zdroj:http://cs.wikipedia.org)

3.3.4. **Evoluční model životního cyklu**

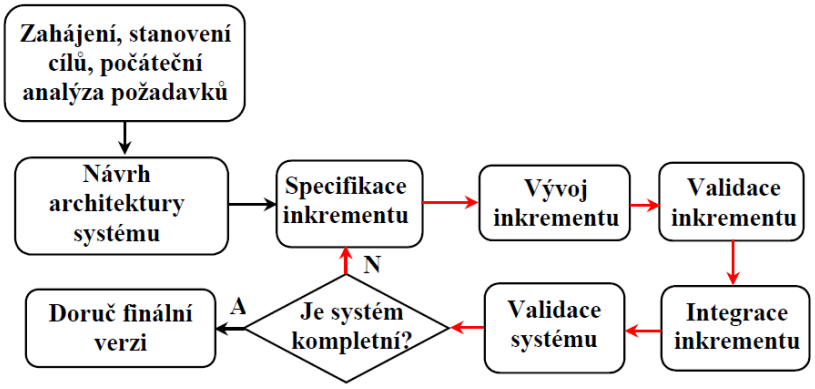
Evoluční model je založen na realizaci projektu postupně, po malých částech, přičemž systém vzniká tak, jak se mění požadavky na něj. Nejznámější metodou je inkrementální model životního cyklu. Tento model v poslední době získal velkou oblibu u komunity OOP. Výhoda tohoto postupu je v možnosti postupného vývoje a údržby a krátké době mezi zadáním požadavku a jeho zapracováním. Nevýhodou jsou opět špatné možnosti sledování a řízení projektu, především u větších projektů. (Merunka, 2008)



Obrázek 3.4 Evoluční model (zdroj:http://katedry.fmmi.vsb.cz)

Inkrementální model životního cyklu

Inkrementální, neboli přírůstkový model životního cyklu si klade za cíl co nejrychlejší dosažení nějakého přínosu v projektu. Tento přístup rozdělí řešený problém do malých celků a řeší každý celek samostatně. (Objektový přístup, 2008)



Obrázek 3.5 Inkrementální model (zdroj: http://www.fit.vutbr.cz)

3.4. Metodiky tvorby IS

Metodika tvorby softwaru nebo také proces vývoje softwaru je nástroj, který nám definuje otázky typu kdo, co, kdy a jak.(Arlow a Neustadt, 2011).

Všechny dnes vytvořené metodologie tvorby softwaru spadají do oboru softwarového inženýrství[2]. Na obrázku Obrázek 3.6 je znázorněn zjednodušený postup tvorby IS.

Obrázek 3.6 Zjednodušený nákres tvorby IS (zdroj: Arlow a Neustadt, 2011)

3.4.1. Historie metodiky tvorby IS

První metodický framework[3] pro vývoj IS byl vytvořen v šedesátých letech minulého století a byl pojmenován SDLC (systems development life cycle), tedy životní cyklus tvorby systémů. Jeho hlavní myšlenkou bylo vytvořit velmi účelný, strukturovaný, metodický nástroj pro vývoj IS, jež by zastřešoval všechny fáze tvorby od nápadu, až po finální dodání systému. Hlavním cílem tehdejšího frameworku bylo vytvořit obsáhlé funkční obchodní systémy. Tehdejší IS nabízely hromadné zpracování dat a aritmetické výpočty. (Elliott, 2004)

---

[2] Obor, který je zaměřený na metodiky a nástroje používaných během životního cyklu softwarových aplikacích. (Fakulta informačních technologií, 2012)

[3] Framework je ucelený soubor knihoven zaměřených na určité téma, které pomáhají tvůrcovi v práci. (Co je to framework?, 2010)

3.4.2. **Kategorizace metodik IS**

Vzhledem ke skutečnosti, že existuje velké množství metodik, které nejsou nijak kategorizovány a ani nějakým způsobem popsány, vzniká problém jaký typ metodiky vybrat na daný typ projektu. (Buchalcevová, 2005).

V následujících bodech jsou rozepsané důvody, proč jsou známy metodiky v tak velkém počtu.

1. Různé technologie vyžadují různé techniky a metody.
2. Organizace se liší firemní kulturou. Odlišná firemní kultura
3. Každý jedinec je jedinečný.
4. Každý tým je jedinečný.
5. Projekty se liší velikostí týmu.
6. Projekty se liší svou důležitostí.
7. Projekty se liší podle postavení produktu na trhu.
8. Projekt existuje v rámci určitého specifického vnějšího prostředí.

Jak dále Buchalcevová uvádí, tyto skutečnosti mají za následek existenci mnoha metodik. Metodiky se liší například podle životního cyklu, který postihují nebo jakým způsobem definují jednotlivé kroky budování informačního systému. (Buchalcevová, 2005)

Pro určení správné metodiky je zapotřebí zvolit správná kritéria a podle nich zvolit správnou metodiku.

3.4.3. **Kritéria metodik budování IS**

1. **Zaměření metodiky:**

   Základním rozdílem je, zda je IS zaváděno do celého podniku, nebo pouze na určitou část.

   a. Globální metodiky – zaměření na celopodnikové IS.
   b. Projektové metodiky – zabývající se pouze jedním projektem, například zavedení IS do části podniku.

2. **Rozsah metodiky:**

   Rozsahem zde chápeme především počet fází životního cyklu IS, které metodika zahrnuje.

a. Celý životní cyklus vývoje.
b. Dílčí metodiky – pouze část životního cyklu IS, např. jen analýzu a návrh.

3. **Váha metodiky**

   Třídí metodiky dle ukazatelů zvaných zkratkou PARTS (precision, accuracy, relevance, tolerance, scale). Na základě těchto ukazatelů vznikají dvě základní třídy:

   a. těžké metodiky – jde o propracované metodiky s přesně definovanými činnostmi a procesy, též zvané rigorózní metodiky,
   b. lehké metodiky – definice principů a praktik, místo podrobných procesů.

4. **Přístup k řešení**

   Metodiky, jež lze uplatnit pouze na nový vývoj IS, který zohledňuje, jakým způsobem bude IS vytvářen:

   a. strukturované metodiky,
   b. objektově orientované metodiky,
   c. metodiky pro komponentový vývoj,
   d. metodiky pro vývoj orientovaný na služby.

5. **Typ řešení**

   Dnes zpravidla neprobíhá vývoj softwaru od začátku, ale často se rozšiřují stávající řešení:

   a. vývoj nového řešení (na zelené louce),
   b. integrace řešení,
   c. rozvoj a rozšíření řešení (upgrade),
   d. customizace a implementace typového řešení,
   e. užití řešení.

6. **Doména**

   Jde o oblast, pro kterou se IS vyvíjí. Datový sklad bude jiný IS než CRM systém apod. (Buchalcevová, 2005)

Tyto kritéria je možné uplatnit na jakoukoliv metodiku vývoje IS, nebo samozřejmě na jakýkoliv vývoj softwaru.

Pokud se budeme bavit o hlavním rozdělení metodik, rozlišujeme dva hlavní proudy. Rigorózní metodiky a agilní metodiky. Tyto metodiky jsou rozděleny především kritériem: váhy metodiky, viz výše. (Buchalcevová, 2005)

Následující dvě podkapitoly objasňují rigorozní a agilní metodiky, které by mohly být použity pro tvorbu IS ve vybraném podniku.

3.4.4. **Rigorózní metodiky**

V této kapitole jsou popsány rigorózní metodiky vývoje softwaru. Charakteristikou těchto metod je, že procesy při budování softwaru lze popsat, řídit, plánovat a měřit. Snaží se o podrobný popis procesů, činností a vytvářených produktů. To je důvod, proč tyto metodiky bývají často objemné. Tyto metodiky bývají často založeny na sekvenčním (vodopádovém) cyklu vývoje, ale existují i rigorózní metodiky založené na interaktivním a inkrementálním vývoji. (Buchalcevová, 2005)

Hlavní principy rigorózních metodik

- požadavky je možné specifikovat předem
- změnám se snažíme zabránit
- hodně dokumentace (Buchalcevová, 2005)

Typy rigorózních metodik

- **Unified Process - UP**

  Metodika Unifield Process je rigorózní metodika, která spojuje iterativní a inkrementální životní cyklus vývoje. Je rozdělena do 4 částí (zahájené, rozpracování, kontrukce, zavedení), kde se vždy provádí sběr požadavků, analýza, návrh, implementace a testování. Tato metoda je řízena případy užití a riziky. Navíc metodika obsahuje podrobný popis použití jazyka UML.

- **Rational Unified Process - RUP**

  Jde o komerční metodiku UP, která byla vytvořena firmou Rational. Metodiky jsou si velice podobné. Rozdílné jsou především v různých detailech a úplnosti metodiky RUP. (Arlow a Neustadt,2011).

- **Object–oriented Process, Environment and Notation - OPEN**

  Metoda zaměřená na celý životní cyklus softwaru. Její zaměření je především na procesy. Její hlavní zaměření je na komponentový a objektově orientovaný vývoj aplikací. (Buchalcevová, 2005)

3.4.1. Agilní metodiky

„Změny technologií a ekonomického prostředí, ke kterým v současnosti dochází, a požadavky na rychlé zavedení IS/ICT vyžadují změny v metodikách. Tradiční rigorózní metodiky přestávají v takových podmínkách vyhovovat a začínají se prosazovat metodiky, které umožňují vytvořit řešení velmi rychle a pružně jej přizpůsobovat měnícím se požadavkům. Tyto metodiky jsou označovány jako agilní.“ (Buchalcevová, 2005)

Hlavní principy agilních metodik:

„Manifest uvádí: Odhalili jsme lepší způsob vývoje software, sami jej používáme a chceme pomoci i ostatním, aby jej používali.“ (Buchalcevová, 2005)

Manifest agilního vývoje softwaru udává čtyři základní hodnoty, kterými se při vývoji softwaru je důležité řídit, přičemž levá část tvrzení je důležitější než pravá strana.

- individualitám a komunikaci - před procesy a nástroji,
- provozuschopnému software - před obsažnou dokumentací,
- spolupráci se zákazníkem - před sjednáváním kontraktu,
- reakci na změnu - před plněním plánu. (Buchalcevová, 2005)

Buchalcevová uvádí v další publikaci čtyři základní pravidla, kterými se agilní metodiky řídí:

- nepopisují procesy, ale soustředí se na praktiky a principy,
- minimum dokumentace, upřednostnění osobní komunikace,
- soustředí se na činnosti vytvářející hodnotu,
- blízká spolupráce zákazníka s vývojářem. (Buchalcevová, 2010)

Typy agilních metodik

- **Extrémní programování - XP**

  Metodika založená na čtyřech základních principech, kterými jsou komunikace, jednoduchost, zpětná vazba a odvaha. Tato metodika není cílená na zápis všech

požadavků, právě naopak. Vždy se soustředí na vytvoření základního programu, který splňuje aktuální požadavky. (Extrémní programování v praxi, 2002)

- **Open Unified Process - OpenUP**
  Stejně jako UP a RUP v rigorózních metodách je tato metoda řízena případy užití a riziky. Tato metodika je kombinaci agilního přístupu a metodiky UP. (Introduction to OpenUP, 2007)
- **Feature–Driven Development - FDD**

  Metodika je postavena na iterativním životním cyklu vývoje. Projekt je řízen tzn. užitnými vlastnostmi, což je malý výsledek, užitečný pro zákazníka. Nejdříve je vytvořen celkový model softwaru. Poté se pokračuje po čtrnácti denních iteracích, ve kterých probíhá vždy návrh a následná realizace jedné užitné vlastnosti. (Buchalcevová, 2005)

- **Scrum**

  Metodika určená pro malé a zkušené týmy. Metodika je založena na krátkých iteracích tzn. sprintech (doba tvrání cca 14 dní). Na začátku se vždy stanoví cíle a na konci se předvádí část funkční aplikace zákazníkovi.(Aspect Works, 2012)

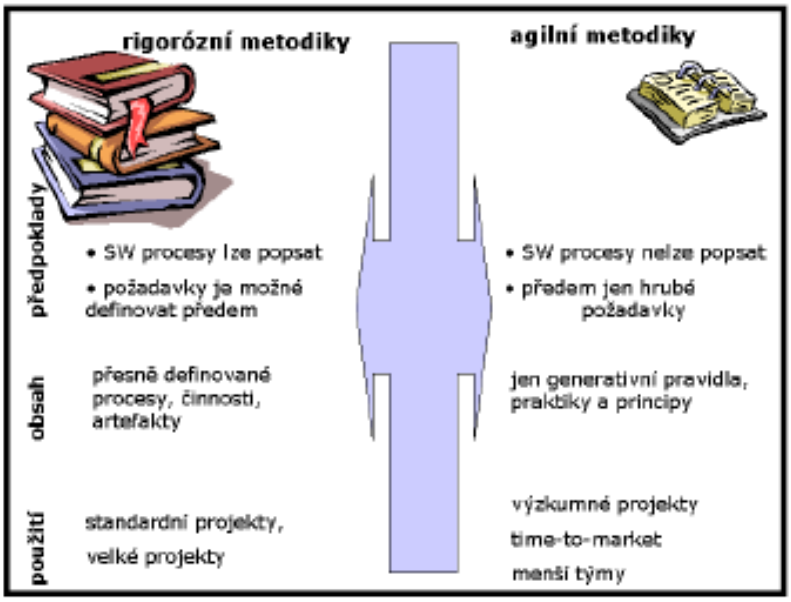
3.4.2. Porovnání rigorózních a agilních metodik

„Příčina vzniku agilních metodik je samotnými autory popisována jako selhání rigorózních metodik. Příčinu jejich selhání vysvětlují v nepotřebné dokumentaci, zbytečném kreslení diagramů a vůbec v provádění aktivit, které oddalují „opravdové“ programování. Jako lék nabízejí tyto zdržující aktivity nedělat vůbec a soustředit se jen na vlastní tvorbu softwaru.“ (Merunka, 2008)

Dále autor Merunka uvádí, že s tímto tvrzením nelze úplně souhlasit a říká, že v každém odvětví (stavebnictví, strojírenství aj.) je potřeba analyzovat, dokumentovat, plánovat atd. Proč by to tedy ve vývoji softwaru mělo být jiné? (2008).

Obdobně problém popisuje Buchalcevová, která říká, že tyto metodiky vycházejí z odlišných předpokladů a odlišného pohledu na software. Například agilní metodiky nejsou uzpůsobeny na údržbu a inovace.(2005)

Na obrázku Obrázek 3.7 jsou zachyceny zásadní rozdíly mezi metodikami.



Obrázek 3.7: Srovnání rigorózních a agilních metodik (zdroj: Buchalcevová, 2005)

3.5. Metodika UP

Následující kapitola je věnována metodice Unified Process (dále jen UP). Byla vybrána metodika z řady rigorózních metod z důvodu velikosti projektu, znalosti všech požadavků předem a požadavku ze strany zadavatele na menší zatížení zaměstnanců firmy při tvorbě nového IS.

Metodika UP je použita v kapitole vlastního zpracování. Tato metoda je zvolena z důvodu otevřenosti standardu pro všechny, na rozdíl od jiných, například RUP, který je vázán na produkty a dodavatele.

Velkou výhodou metodiky UP je, že má plnou podporu nástrojů UML. Více o jazyce UML v kapitole 3.6.

3.5.1. Historie UP

Kořeny metodiky UP sahají až do roku 1967, kdy vznikl Ericssonův model, jehož pojetí vycházelo z faktu, že složité systémy se musí modelovat jako množiny vzájemně propojených bloků. V roce 1987 byla založena firma Objectory AB panem Jacobsonem, vytvářející metodiku Objectory. Tato metodika se skládá z množiny dokumentací nástroje CASE. Tato metodika i přes řadu šablon pro různé projekty musela být takřka vždy uzpůsobena pro zákazníka. V roce 1995 koupila firmu Objectory AB společnost Rational,

která se snažila o sjednocení metody Objectory s metodami firmy Rational. Výsledkem byla architektura pohledů 4+1 (logickém, procesním, fyzickém a vývojovém) a jedním, který je všechny sjednocoval a to pohledem případů použití. Tento pohled je i dnes základem UP. Iterativní vývoj byl formalizován do čtyř fází, a to: zahájení, rozpracování, konstrukce a zavedení. Tyto fáze obsahují vodopádový životní cyklus s dynamikou inkrementálního vývoje. Výsledkem tohoto spojení vznikla metoda ROP (Rational Objectory Proces). V této době v této firmě vznikal i nový jazyk UML (Unified Modeling Language). V roce 1997 firma Rational skupuje spoustu firem a získává tak mnoho znalostí v oblastech sběru požadavků, správě konfigurace, testování adt. Díky tomuto pokroku v roce 1998 vzniká nová metodika RUP. Od této doby vzniká mnoho podobných metodik. Roku 1999 byla publikována kniha Unified Software development Process (Unifikovaná metodika vývoje softwaru), v níž je detailně popsaná metodika UP. Zatímco RUP je produkt firmy Rational, tak UP je otevřeným standardem. (Arlow a Neustadt, 2011).

3.5.2. **Axiomy metodiky UP**

Metodika UP obsahuje tři základní axiomy, kterými jsou:

1. zásada řízení případem užití a rizikem,
2. zásada soustředění se na architekturu,
3. zásada iterace a přírůstků

Řízení případem užití a riziky

Lze říci, že celá metodika UP je řízena případy použití, podrobnější vysvětlení v kapitole 3.7.6. Při řízení projektu probíhá navíc analýza možných rizik. Tuto práci řeší manažer projektu.

Zásada soustředění se na architekturu

UP pro tvorbu softwaru je založena na návrhu a postupném vývoji architektury daného systému. Architektura popisuje jak strategické aspekty, tak jakým způsobem se rozkládá systém na jednotlivé komponenty a komunikaci mezi nimi. Dnes je již jasné, že velké systémy jsou postaveny na dobré architektuře a že není rozumné vymýšlet tyto projekty z patra s minimální znalostí budoucího kódu.

Zásada iterace a přírůstků

Iterativní znamená, že projekt je rozdělen do menších podprojektů, které jsou řešeny samostatně. Přírůstkové zde znamená, že je projekt tvořen postupným vylepšováním a zpřesňováním daného záměru. (Arlow a Neustadt, 2011).

3.5.3. Iterativní přístup v UP

Historie ukazuje, že je jednodušší řešit věci po malých částech, než celý problém najednou. Z toho vychází i metodika UP. Snaží se tedy rozdělit celý projekt do několika interakcí tzv. „miniprojektů“. Každý z těchto miniprojektů obsahuje vždy prvky normálního projektu, kterými jsou:

- plánování,
- analýza a návrh,
- tvorbu,
- integrace a testování,
- interní nebo externí uvedení.

Každá interakce má svoji linii neboli baseline, která se skládá z částečně kompletní verze a z všech přidružených dokumentů. Rozdíl mezi jednotlivými iteracemi je označován jako přírůstek. Proto se tato metoda označuje jako přírůstková. (Arlow a Neustadt,2011).

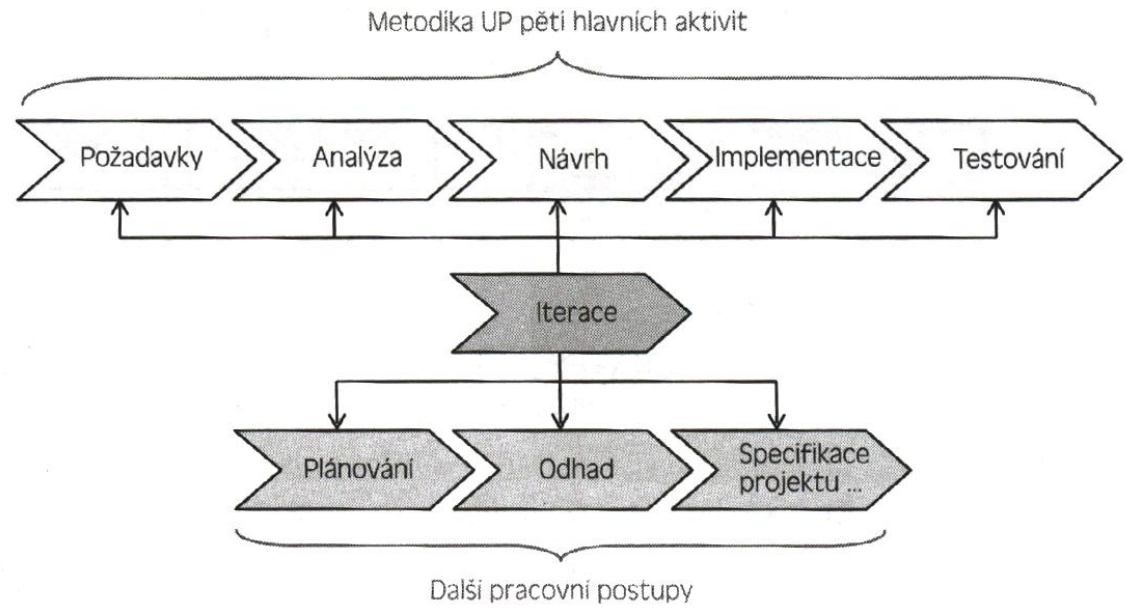
3.5.4. Pracovní postupy v UP

Každá iterace obsahuje pět základních pracovních postupů neboli workflow. Iterace mohou obsahovat i další postupy, jako jsou plánování, odhady atd., ale ty již nejsou součásti metodiky UP. Postupy popsané v metodice jsou:

1. Požadavky – Zachycení co systém má dělat.
2. Analýza – Struktura požadavků.
3. Návrh – Realizace požadavků v architektuře systému.
4. Implementace – Vytvoření systému.
5. Testování – Ověření, zda implementace neobsahuje chyby a je dle požadavků.

Tyto postupy může obsahovat každá iterace. To, jaké obsahuje, závisí na fázi projektu. Rozdělením celého projektu na jednotlivé iterace dosáhneme lepšího plánování celého projektu. Jednotlivé iterace na sebe můžou navazovat, ale můžou běžet i současně. Tím

je možné dosáhnout využití celého týmu hned od začátku projektu a tím i zkrácení doby trvání celého projektu. (Arlow a Neustadt,2011).



Obrázek 3.8: Pracovní postup v metodice UP

3.5.5. **Struktura metodiky UP**

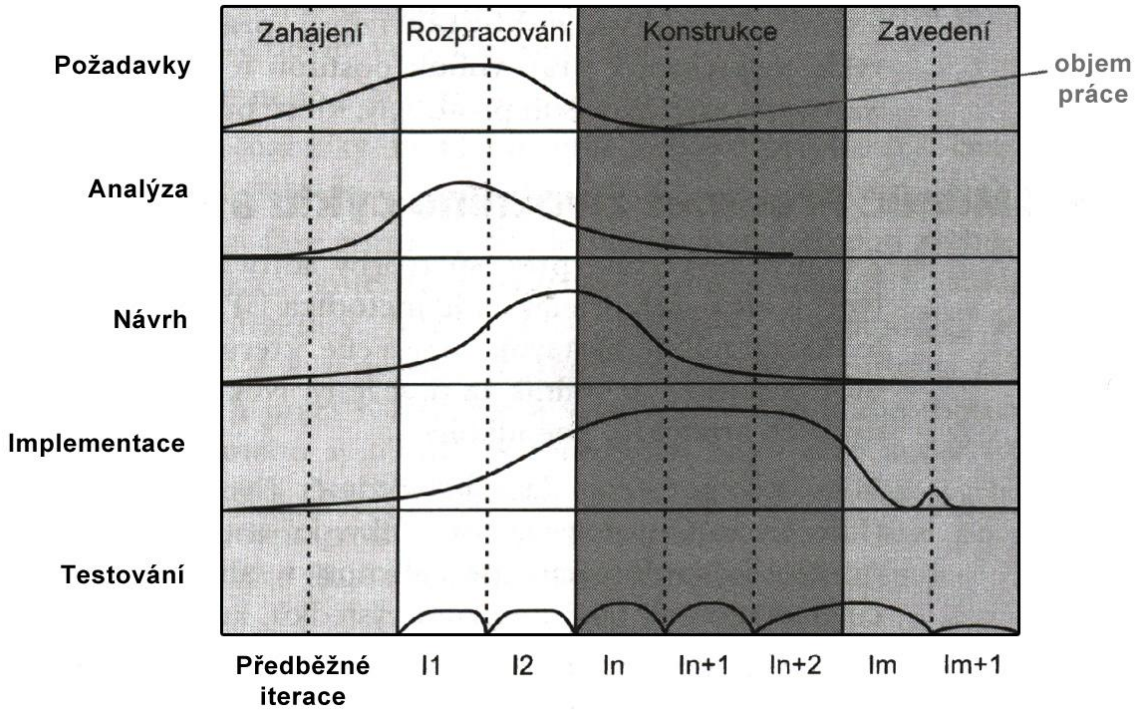
Struktura metodiky UP je rozdělána do 4 čtyř částí, kterými jsou:

1. Zahájení – období plánování.
2. Rozpracování – období architektury.
3. Konstrukce – počátky provozuschopnosti.
4. Zavedení – nasazení produktu do uživatelského prostředí.

Každá z těchto částí může obsahovat jednu nebo více iterací. Počet jednotlivých iterací v dané části se odvíjí od velikosti projektu. (Arlow a Neustadt, 2011).

3.5.6. **Fáze metodiky UP**

Na obrázku Obrázek 3.9 je možné vidět popis jednotlivých fází metodiky UP s propojením na pracovní potupy. Každá z fází plní určitý cíl, viz popis níže.



Obrázek 3.9: Fáze metodiky UP (zdroj: Arlow a Neustadt,2011)

Fáze zahájení

Hlavní cíl této fáze je odstartování projektu. Tato fáze obsahuje tvorbu podmínek proveditelnosti, nadnesení obchodního případu, zachycení podstatných požadavků a označení kritických rizik. Hlavní důraz v této fázi je kladen na pracovní postupy zabývající se specifikací požadavků a jejich analýzou. Tato fáze se nezabývá implementací ani testováním. Každá fáze je ukončena po dosažení jistých cílů. Tyto cíle musí být splněny pro postup do další fáze. Pro každý projekt se tyto cíle mohou lišit podle toho, jaké jsou v daném projektu důležité. (Arlow a Neustadt,2011).

Fáze rozpracování

Cílem této fáze je vytvoření spustitelného základu, vylepšení odhadu rizik, zachycení případu užití, tvorba přesného plánu konstrukční fáze, vytvoření analytického modelu, zachycení stabilní architektury. (Arlow a Neustadt,2011).

Fáze konstrukce

Cíl konstrukční fáze je splnění všech požadavků zachycených v analýze a návrhu a jejich nasazení na vytvořenou architekturu z fáze rozpracování. V této fázi bychom měly mít zachycené všechny požadavky, i když stále ještě mohou vznikat další. Těch by ale mělo být minimum. Dokončuje se analýza a návrh systému a vzniká první funkční

a spustitelný systém, u kterého samozřejmě probíhá i jeho testování. (Arlow a Neustadt,2011).

Fáze zavedení

V této fázi je kladen velký důraz na implementaci a testování. Nevznikají žádné další požadavky na systém a neprobíhá jejich analýza. Co však ještě probíhá je poslední úprava návrhu. (Arlow a Neustadt,2011).

3.6. Unified modeling language UML

Jazyk UML je otevřený průmyslový standard pro vizuální modelování analýzy a návrhu softwarových aplikací, schválený sdružením OMG (Object Management Group). Jde o unifikaci softwarových domén, jako jsou vývojový cyklus, aplikační domény, implementace jazyka a platformy, vývojové procesy a vlastní interní propojení. Velká část tohoto jazyka je používána v celém procesu UP. (Arlow a Neustadt, 2011)

3.6.1. Historie UML

V osmdesátých letech, po vytvoření programovacího jazyka C++, se začalo uvažovat o vytvoření objektově orientovaného grafického jazyka pro návrhy systémů. Převratnou událostí byl přechod Jima Rumbaugha z GE do Rational Softwaer k Grady Boochovi. Cílem bylo sjednotit do té doby velké množství metodik. V roce 1995 vytváří veřejnou prezentaci na téma Ufield Method. Zlomovou událostí se stala koupě společnosti Objectory, kdy se do týmu vývojářů přidal její zakladatel, Jacobson. Roku 1997 několik organizací podpořilo tyto modelové řešení a vzniká první verze UML.(Fowler, 2009)

3.6.2. UML diagramy

V následující tabulce číslo Tabulka 3.1 jsou vypsány jednotlivé diagramy, které obsahuje UML verze 2.

Název diagramu	Popis
Aktivit	Procesní a paralelní chování.
Balíčků	Hierarchická struktura pro překlad.
Časování	Interakce mezi objekty, důraz na časování.
Komponent	Struktura a propojení komponent.
Komunikace	Interakce mezi objekty, důraz na spojení.
Nasazení	Nasazení architektury na uzly.
Objektů	Příklad uspořádání instancí.
Přehledu interakcí	Kombinace sekvenčního diagramu a diagramu aktivit.
Případu užití	Jak uživatelé komunikují se systémem.
Sekvenční	Interakce mezi objekty, důraz na sekvence.
Složení struktur	Dekompozice tříd za běhu programu.
Stavový	Jak události mění objekty během jeho života.
Tříd	Třídy, vlastnosti a vztahy.

Tabulka 3.1 Typy diagramů (zdroj: Fowler, 2009)

3.6.3. Objekty jazyka UML

Základním předpokladem je, že je možné jakýkoliv software popsat pomocí spolupracujících objektů. Přestože toto lze jednoznačně říci o OOP, tato představa zapadá i do obchodních procesů a dalších aplikací. Objekty lze rozdělit do dvou základních částí:

- **Statická struktura –** Popisuje, jakým způsobem jsou tyto objekty propojeny a jak spolu souvisí.
- **Dynamická struktura** – Popisuje, jakým způsobem spolu objekty spolupracují za cílem dosažení určité funkčnosti v daném čase. (Arlow a Neustadt, 2011)

3.6.4. Způsoby použití UML

Při snaze charakterizovat způsoby použití UML se na sobě nezávisle shodli Steve Mellor a Martin Fowler a definovali způsoby použití jako:

- **Tvorbu náčrtků**, které používají vývojáři pro usnadnění interní komunikace. Lze tak používat buďto **dopředné inženýrství**, při kterém se vytváří zdrojový kód (označení pro text, který vytváří funkčnost programu) systému na základě UML, nebo **zpětného inženýrství** u kterého naopak popisujeme za pomocí UML část kódu, například pro lepší pochopení jeho funkčnosti.

Cílem těchto náčrtků je především zlepšení interní komunikace mezi jednotlivými vývojáři. Tyto náčrtky mají často neformální charakter, protože je potřeba rychlého nákresu v kooperaci s kolegy. Je běžné tyto náčrtky kreslit pouze na bílou tabuli.

- **Tvorbu návrhů**, která se používá ke kompletnímu popsání systému, kdy návrhář (designer) připraví detailní návrh systému pro programátora, který jej pouze vezme a zakóduje. Tento návrh by měl být co nejvíce kompletní, aby programátor při jeho kódování musel co nejméně přemýšlet o funkčnosti systému a mohl se pouze soustředit na programování. Opět je možná využít dopředného i zpětného inženýrství.
- **UML jako programovací jazyk**. Při použití výkonných nástrojů je možné ze správně zapsaných diagramů je možné z UML vygenerovat spustitelný zdrojový kód, čímž se stává UML zdrojovým kódem. Zde nemá dopředné a zpětné inženýrství smysl, neboť UML a zdrojový kód je totéž. (Fowler, 2009)

3.7. Sběr požadavků

Jak již bylo výše uvedeno, celá metodika UP je řízena požadavky, proto než je spuštěna samotná analýza, musí být nejdříve jasně určeno, podle čeho bude analýza provedena. Z toho vyplývá, že sběr požadavků je jednou z velmi důležitých částí tvorby systémů. Tomuto sběru požadavků se odborně říká inženýrství požadavků neboli requirements engineering. (Arlow a Neustadt,2011).

Autoři Kanisová a Müller tvrzení, že je sběr požadavků velice důležitý potvrzují a říkají, že je velice důležité tuto oblast nepodcenit, protože zanedbání některých požadavků může vést v pozdějším vývoji k velkým problémům (2006).

S tím lze pouze souhlasit, i když do jisté míry díky inkrementálnímu přístupu UP je tento problém zmenšen, pokud by nešlo o poslední fázi nasazení, nebo by požadavky nezasahovaly do architektury aplikace. Naopak pokud by byl použit vodopádový přístup vývoje softwaru, byl by tento problém obrovský.

3.7.1. Definice požadavků

Dá se říci, že v zásadě každý systém obsluhuje více uživatelů, v metodice UP se jim říká dělníci. Jedním z úkolů sběru požadavků je tedy tyto dělníky najít. Druhá a neméně

důležitá část je vydefinovat všechny požadavky kladené na samotný systém. Tyto požadavky rozdělujeme na funkční a nefunkční.[4]

Funkční požadavky

Jsou požadavky, které zachycují funkcionalitu systému jako takového. Tyto požadavky jsou v pozdější fázi zapisovány do případů užití, viz kapitola 3.7.6.

Ukázka funkčních požadavků:

- Pokladna bude ověřovat platební kartu.
- Pokladna bude požadovat přihlášení před jakoukoliv akcí.

Nefunkční požadavky

Naopak nefunkční požadavky říkají, jaké omezení systém bude mít.

- Pokladní systém bude napsán v jazyce JAVA.
- Pokladní systém bude odpovídat max. do 0,5 sekundy.

Formulace požadavků

Při soupisu požadavků je dobré uchýlit se k jednotnému zápisu s jasnou identifikací každého jednotlivého požadavku. Vývojáři často doporučují velice jednoduchý zápis požadavků ve tvaru:

<id><systém> bude <funkce>

Kde <id> je jasný identifikátor požadavku, označení <systém> označuje vykonávaný systém a <funkce> říká, jaké funkcionalita bude provedena.

Uspořádání požadavků

Pro rozsáhlejší projekty je vhodné rozdělit funkční a nefunkční požadavky ještě do dalších kategorií pro lepší orientaci. Rozřazení může být libovolné dle projektu, nebo zvyku pracovního týmu. Toto rozdělení se většinou používá u sto a více požadavků.

---

[4] Požadavky je možné kategorizovat i jiným rozsáhlejším způsobem, avšak pro zjednodušení v této práci jsou rozlišeny pouze na funkční a nefunkční požadavky.

Atributy požadavků

Každý požadavek může obsahovat libovolný počet rozšiřujících informací, tzn. metadat. Snad nejběžnějším je atribut priorita, která udává důležitost požadavku. V tabulce číslo Tabulka 3.2, jsou definovány jednotlivé hodnoty pro tento atribut. (Arlow a Neustadt, 2011)

Hodnota atributu priorita	Sémantika
Nezbytný (must have)	Povinné požadavky, jež jsou základem systému.
Možný (should have)	Důležité požadavky, které však lze vynechat
Eventuální (could have)	Požadavky nepovinné.
Chceme mít (want to have)	Požadavky, které můžou být v dalších verzích

Tabulka 3.2 Atribut priorita požadavku (zdroj: Arlow a Neustadt, 2011)

3.7.2. Zdroje požadavků

Proces získávání požadavků je náročná činnost. Je možné setkat se s uživateli, kteří budou mít jasnou představu o celém systému a dokážou jasně formulovat požadavky a služby. Naopak je možné narazit na uživatele, kteří přenechají veškerou aktivitu na tvůrce systému a požadují dodání aplikace, která bude přesně podle jejich představ, aniž by pomohli s upřesněním požadavků. Jako zdroje můžeme identifikovat:

- Legislativu.
- Požadavky zákazníků.
- Existující systém zákazníků.
- Pracovní procesy zákazníků.
- Vlastní know-how pro danou problematiku.
- Prostředí zákazníka.
- Hardware a software zákazníka. (Kanisová a Müller, 2006)

3.7.3. Získávání požadavků od zákazníka

„Kdykoliv při zachycení požadavků na softwarový systém pracujete s lidmi, snažíte se jejich prostřednictvím získat co nejpřesnější obraz nebo mapu pracovního modelu.“ (Arlow a Neustadt, 2011, 84)

Při sběru požadavků je vždy důležité si uvědomit, zda podaná informace již není obsažena v jiném požadavku, zda je opravdu pravdivá a zda je již dostatečně podrobná. (Arlow a Neustadt, 2011)

Konzultace

Jde o nejběžnější, nejpřirozenější a nejpříjemnější metodu sběru požadavků. Je důležité sbírat požadavky jednotlivě a vždy pouze od zainteresovaných osob v systému. Každý, kdo sbírá tímto způsobem požadavky, by měl mít na paměti tyto pravidla:

- Nemít představu o systému.
- Zadávat otázky bez kontextu.
- Nechat zákazníky samostatně povídat.
- Pečlivě naslouchat a nesnažit se dokončovat věty.
- Mějte trpělivost.

Dotazník

Dotazníky mohou být užitečné pro doplnění konzultace, ale není vhodné je používat jako samostatnou metodu pro sběr požadavků.

Dílna požadavků

Jde o nejefektivnější metodu pro zachycování požadavků, zaměřenou především na nové požadavky. Jedná se o metodu, kdy se v jedné místnosti sejdou sparingpartneři, inženýři požadavků, zástupci zákazníka a experti v oboru. Začíná se spontánní diskuzí na téma zvoleného projektu. Všechny požadavky se ihned zapisují (nevyvrací se ani se o nich nediskutuje). Později každý z členů vyjmenuje nejdůležitější požadavky na systém. Tyto požadavky se napíší na lísteček a nalepí na tabuli, nebo nástěnku, aby je každý viděl. Později po skončení schůzky se všechny požadavky analyzují. Toto setkání se může opakovat několikrát, dokud není vytvořena zcela jasná představa o funkcionalitě systému. (Arlow a Neustadt, 2011)

3.7.4. Cíl sběru požadavků

Hlavním cílem v metodice UP při sběru požadavků je vyhledání a stanovení následujících bodů:

- Vyhledání aktérů a případů použití.
- Detail případů použití.
- Struktura modelu případu použití. (Arlow a Neustadt, 2011)

3.7.5. **Slovník pojmů**

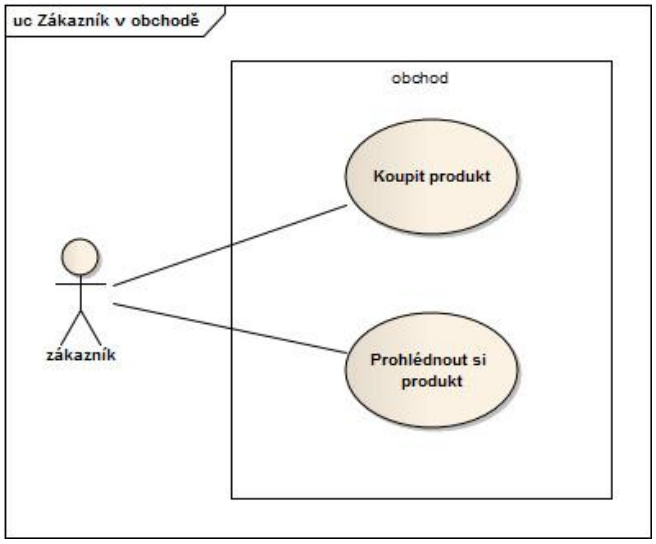
Slovníček pojmů je jednou z nejdůležitějších částí. Obsahuje žargon daného odvětví a vypovídá o jednotlivých pojmech, které se v projektu vyskytují. Každý slovníček by měl obsahovat synonyma[5] a homonyma[6] pro každý pojem. (Arlow a Neustadt, 2011)

3.7.6. **Případy užití**

Modelování případů užití je jednou ze základních prací inženýrství požadavků. Hlavní cíl při tomto modelování je nalezení hranice systému, jeho aktérů a jednotlivých případů užití. (Arlow a Neustadt, 2011)

Hlavní komponenty případů užití jsou:

- **Hranice systému** – ohraničení případu užití.
- **Aktéři** – role zastávaná v systému.
- **Případy užití** – úlohy, která je schopen aktér vykovávat.
- **Relace** – vztahy mezi jednotlivými aktéry a případy užití. (Arlow a Neustadt, 2011)



Obrázek 3.10 Ukázka jednoduchého případu užití (zdroj: tvorba autora)

[5] Slova stejného nebo velmi podobného významu. (ABZ.cz, 2006)
[6] Slova stejně znějící, ale jiného významu. (ABZ.cz, 2006)

Případy užití

Jde o grafické znázornění zachycení funkčních požadavků na systém. Případy užití popisují interakci uživatele ze systému. Případy užití jsou napojeny na aktéry, čímž je určeno, co aktér může v systému dělat.

Je třeba rozlišení případu užití a diagramu případu užití. Pojmem případ užití se označuje jedna interakce zachyceného požadavku na systém. Naopak diagramem případu užití se rozumí jedna skupina případů užití a aktérů zakreslená v celém diagramu.

Scénáře

Jsou posloupnost kroků jsoucích po sobě k dosažení tíženého výsledku. Scénář by měl zachytit komunikaci mezi uživatelem a systémem. (Fowler, 2009)

Aktéři

Znázorňují jednoho či skupinu reálných uživatelů. Aktéři znázorňují, jakou bude mít uživatel funkcionalitu v budoucím systému.

Vazby

Klasickým spojením je asociace mezi aktérem a případem užití, které určuje, co může aktér v systému dělat. Krom tohoto existují ještě další 3 druhy vazeb:

- relace include,
- relace extend,
- generalizace případů užití.

Relace include

Toto spojení se používá tehdy, pokud jeden případ užití má závislost a nemůže existovat bez druhého. Používá se tehdy, pokud je nějaká část programu, která se často opakuje, například vyhledání uživatele. Tento případ užití se vyskytuje při funkcích, jako je upravení uživatele, smazání uživatele a další. Proto se vytvoří případ vyhledat uživatele a připojí se ke všem případům, které bez něho nemůžou fungovat. Relaci include je značena v UML jako „<<include>>“.

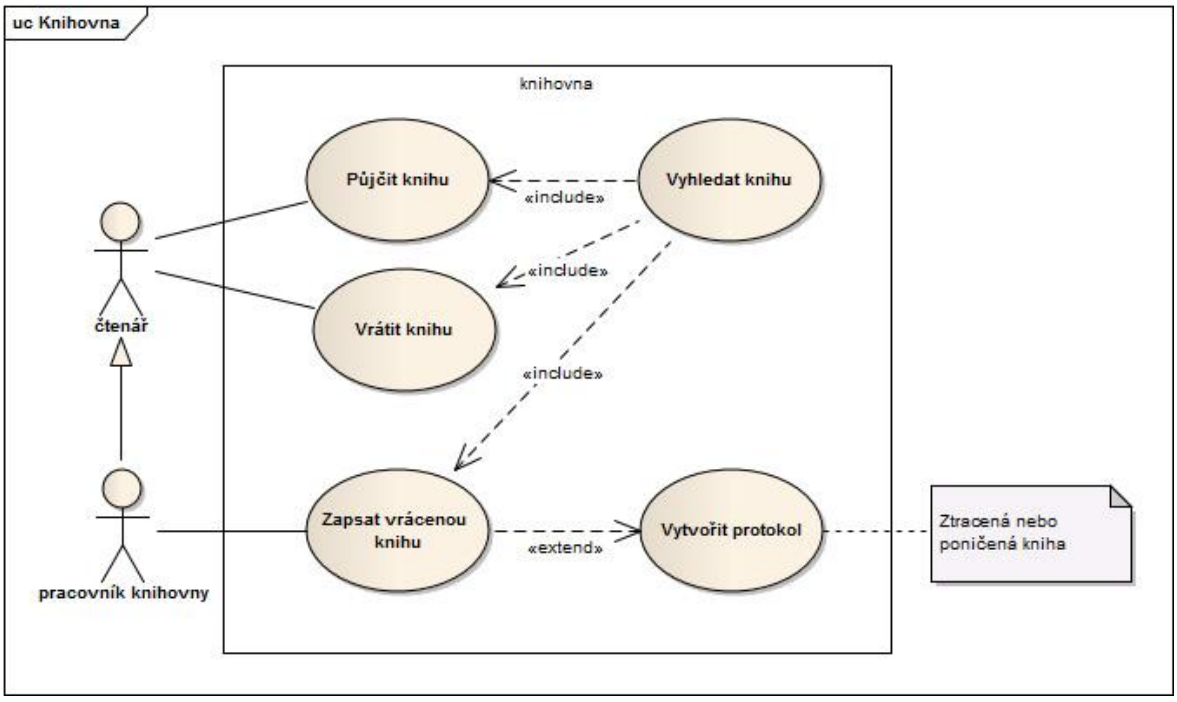
Relace extend

Na rozdíl od relace include, extend rozšiřuje případ užití, na který je napojen a nemusí probíhat, například zaplacení účtu v restauraci, kdy při zaplacení může, ale nemusí být vytištěn účet. Relaci extend je značena v UML jako „<<extend>>“.

Generalizace případů užití

Generalizace případu užití znamená dědění veškeré funkcionalitu od jiného případu. Tento případ se často v diagramu případu užití nevyskytuje. Častější je dědění u aktérů, kdy aktér zdědí možnost používat všechny případy, které může aktér ovládat. Dědění se značí prázdnou šipkou. (Fowler, 2009) (Arlow a Neustadt, 2011)

Na následujícím obrázku jsou znázorněny všechny základní možnosti práce s případy užití.



Obrázek 3.11 Ukázkový případ užití (zdroj: tvorba autora)

3.8. Analýza

Analýza se ve většině případů realizuje na konci fáze zahájení. U fáze zahájení se jedná spíše o požadavky. Ve fázi rozpracován se pak analýza týká především tvorby modelů. Hlavním záměrem analýzy z pohledu objektově orientovaného analytika je především tvorba analytického modelu. Hranice mezi analýzou a návrhem jsou často špatně definovatelné a často záleží pouze na úsudku analytika.

Hlavním cílem analýzy je vydefinování základních tříd systému a jejich propojení na případy užití.(Arlow a Neustadt, 2011)

3.8.1. **Cíl analýzy**

Výstupem z analýzy jsou hlavní dva artefakty. Těmi jsou:

- Analytické třídy.
- Realizace případů užití.

3.8.2. **Třídy**

„Třída je definována jako deskriptor množiny objektů, které sdílejí stejné atributy, operace, metody, relace a chování.“. (Arlow a Neustadt, 2011)

Dá se tedy říci, že třída je obecná množina, která obsahuje objekty vykazující stejné vlastnosti a chování.

Objekty

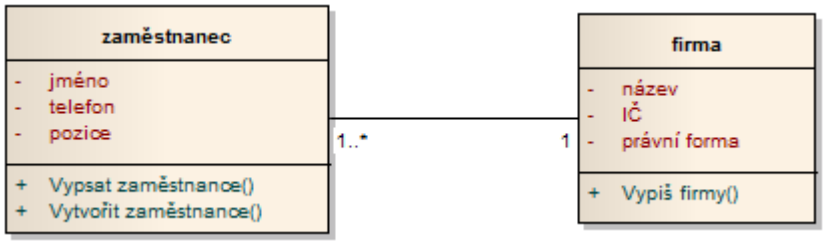
Objekt je konkrétní člen třídy, který již vlastní jasné hodnoty atributů. Příkladem je žák ve škole. Žák se jménem Marek Novotný je konkrétní objekt třídy student školy. (Arlow a Neustadt, 2011)

Atributy

Každá třída obsahuje atributy, které popisují určité vlastnosti dané třídy. Například pokud existuje třída studenti, obsahuje atributy typu jméno, příjmení, třída, studijní průměr atd..

Operace

Uvádí, jaké operace je možné provádět s danou třídou. Například, pokud opět bude třída studenti, operace nad touto třídou budou vypsat studenta, smazat studenta, upravit údaje atd. (Arlow a Neustadt, 2011)



Obrázek 3.12 Jednoduché zobrazení tříd, jejich atributů a operací (zdroj: tvorba autora)

Generalizace - dědění

Generalizace funguje na stejné bázi jako případy užití. Třída, která je ve vztahu generalizace k jiné třídě, dědí všechny její atributy a operace.

Pokud je potřeba použít třídu, která slouží pouze pro účely obecné, tedy neobsahuje žádné objekty, ale od této třídy dědí vlastnosti jiné třídy, nazývá se tato třída abstraktní.

Polymorfizmus

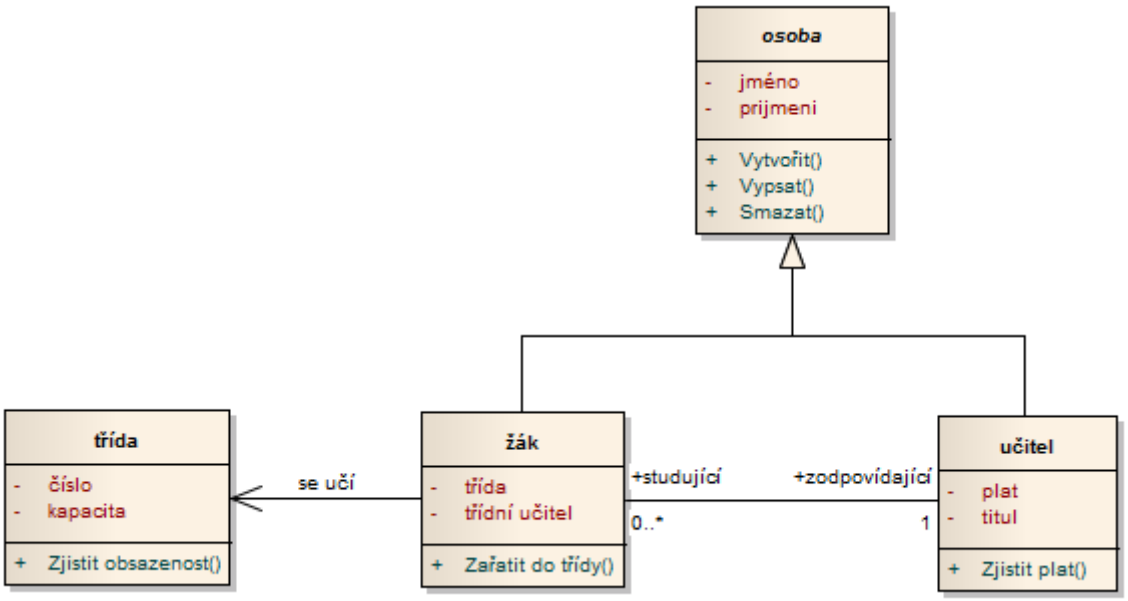
Polymorfizmus znamená mnohotvárnost. V praxi jde o dědění jedné operace pro své potomky. Pokud existuje třída tvar, která obsahuje operaci „kreslit“ a od této třídy dědí další třídy, například třídy kruh a čtverec, operace „kreslit“, je provedena u každé třídy s jiným výstupem, ale pokaždé se nakreslí nějaký tvar. Lze tedy říci, že polymorfizmus je jeden druh operace, který může mít mnoho forem.

Asociace

Asociace popisuje spojení mezi třídami. Vysvětlení je jednoduché, pokud existuje spojení mezi objekty, musí existovat i spojení mezi třídami. Asociace má následující vlastnosti.

- Název – popisuje vazbu mezi třídami, tento název je umístěn uprostřed mezi třídami.
- Název role – je umístěna na straně třídy, ke které se vztahuje a popisuje role třídy ve vztahu tříd.
- Násobnost – udává, kolik objektů z jedné třídy se může vázat na objekty druhé třídy.
- Průchodnost – udává směr, jakým se řídí vztah tříd. (Arlow a Neustadt, 2011)

Na obrázku Obrázek 3.13 jsou zobrazeny všechny vlastnosti asociace, ukázka dědění a abstraktní třídy. Abstraktní třída se značí kurzívou v názvu třídy. (Fowler, 2009).



Obrázek 3.13 Komplexní ukázka diagramu tříd (zdroj: tvorba autora)

Existují i další zpřesňující vazby mezi třídami, ale ty se zavádějí až v návrhu.

3.8.3. **Analytické třídy**

Analytické třídy slouží k jednoznačnému zmapování nějakého obchodního případu, jako je zákazník, produkt nebo účet. Analytické třídy zobrazují jednoznačnou strukturu tříd a jejich základní popis. Při fázi vytváření návrhových tříd ve fázi návrhu jsou analytické třídy upřesňovány. V některých případech se může stát, že z jedné analytické třídy vznikne více návrhových tříd. Navíc v analytických třídách se nezobrazují podrobné popisy operací, jako jsou návratové hodnoty, nebo podrobný popis metod.

3.8.4. **Realizace případů použití**

Pro zachycení případů užití, neboli zobrazení interakce mezi analytickými třídami a případy užití je možné použít různé druhy diagramů interakce.

Diagramy interakce:

- Sekvenční diagramy.
- Komunikační diagramy.
- Diagramy zjednodušené interakce.
- Časové diagramy.

Pro účely analýzy vytvořené v praktické části se nejvíce hodí diagram sekvenční, který umožňuje zobrazit časovou posloupnost při komunikaci jednotlivých tříd. (Fowler, 2009).

3.8.5. **Sekvenční diagram**

„Sekvenční Diagramy znázorňují interakce mezi čárami života jako časově uspořádané posloupnosti událostí. Tyto diagramy jsou nejbohatší a nejpružnější formou diagramů interakce.“ (Fowler, 2009).

Čáry života a zprávy

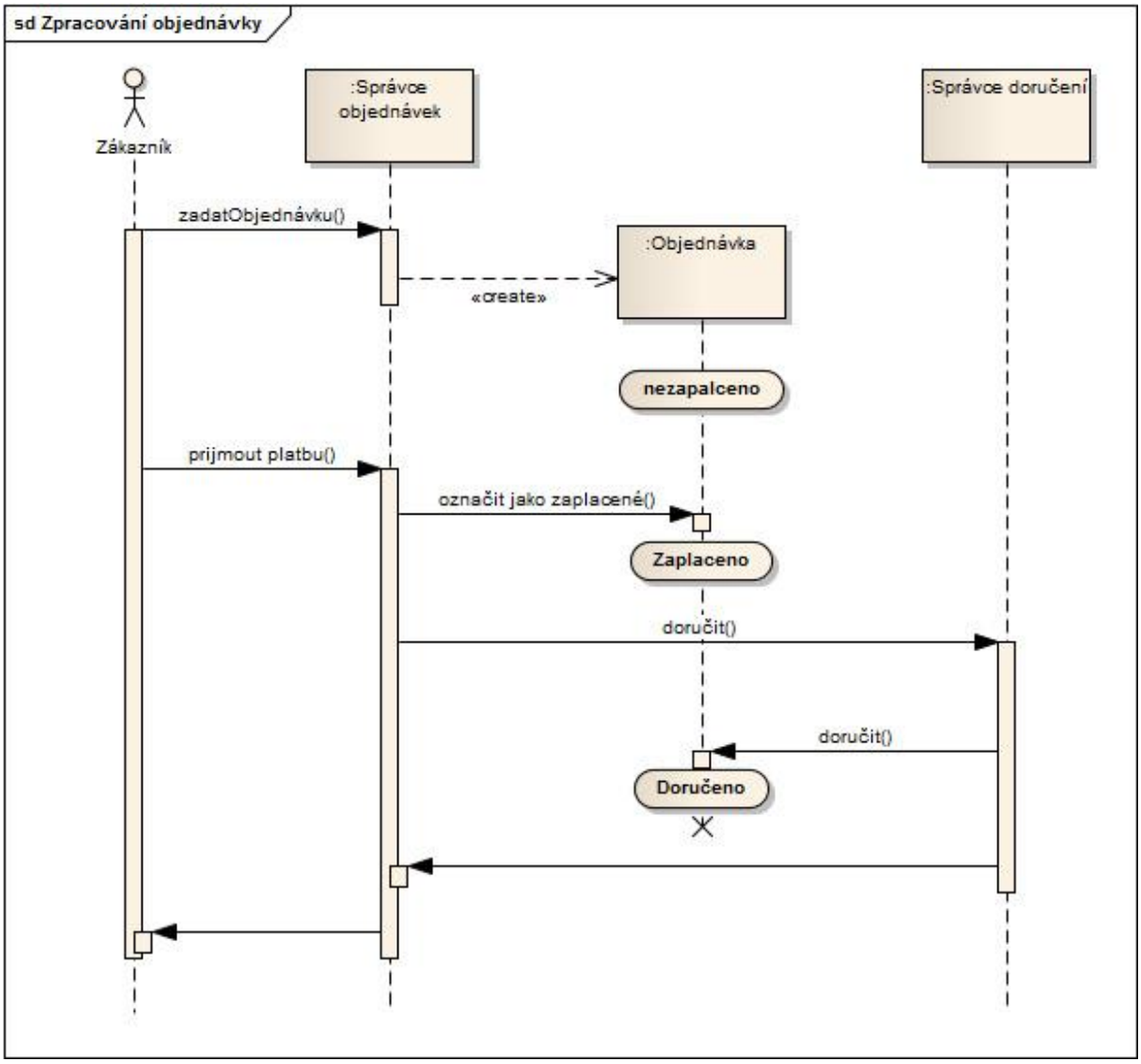
Na obrázku číslo Obrázek 3.14 je předvedena implementace smyšleného scénáře. Sekvenční diagram vytváří u každého účastníka či třídy svislou čáru, neboli čáru života, která znárodňuje časovou posloupnost a šipky mezi těmito čarami, které se nazývají zprávy. Za pomoci zpráv probíhá komunikace mezi objekty.

Cykly a podmínky

Sekvenční diagramy dokážu zobrazit cykly nebo podmínky. Ty jsou zobrazeny pomocí rámců, které se kreslí přes všechny objekty, čáry života a zprávy, pro které jsou platné. V levém horním rohu mají upřesněný název funkce (loop pro cyklus a alt pro podmínku).

Vytváření a rušení účastníka

Při komunikaci mezi objekty je možné zakreslit vytvoření či zánik objektu. Pro vytvoření objektu slouží šipka, jejíž ukazatel směřuje přímo do nového objektu. Naopak zánik se značí velké X na konci životní čáry. (Fowler, 2009).



Obrázek 3.14 Ukázkový sekvenční diagram (zdroj: Arlow a Neustadt, 2011)

3.9. Návrh

K největšímu rozšíření návrhu dochází ve fázi rozpracování. Návrh nám popisuje přesnou definici, jak bude systém fungovat do poslední části. V této části je velice důležité rozhodnout se, zda je nutné udržet si dva modely systému, a to analytický a návrhový. To s sebou přináší spoustu problémů: pokud se předělá analytický model na návrhový, ztratí se abstraktní pohled na systém. Ale naopak pokud se udrží oba modely, je také nutné udržet oba modely aktuální. Proto metodika UP doporučuje mít jeden tým nebo osobu pro návrh a analýzu systému namísto dvou oddělených týmů.

Jedním z příkladů pro lepší pochopení mezi návrhem a analýzou je poukázání na diagram tříd. V analýze jsou třídy popsány velice stroze. Atributy nemají označení typů

proměnných, nebo metody nejsou rozebrány do detailů (jaké vstupují proměnné, jaké vystupují). Také se v analýze spokojíme s obecným popisem metody, jakou třída nabízí, jako je například vypočítaný úrok. V návrhu je již nutné upřesnit jaké další metody bude tato metoda obsahovat, aby dosáhla opravdu vypočítaného úroku. (Arlow a Neustadt, 2011)

V návrhu se především vytvářejí tyto diagramy UML:

- Návrhové a implementační třídy.
- Sekvenční diagramy.
- Diagramy balíčků.
- Stavové diagramy.
- Diagramy nasazení. (Fowler, 2009)

3.10. **Implementace**

Pracovní postup implementace nabízí analytikům a návrhářům již velice málo. Jde o fázi, v které se převádí návrhový model do spustitelného kódu. Systém je z největší části implementován ve fázi konstrukce.

Součástí implementace je i nasazení vytvořeného softwaru u klienta. K tomu slouží diagram nasazení, který popisuje postup nasazení na daný hardware.(Arlow a Neustadt, 2011)

3.11. **Testování**

Testování probíhá takřka v každé fázi krom fáze zahájení. Jde vždy o otestování vytvořených jednotlivých částí spustitelného kódu. (Arlow a Neustadt, 2011)

3.12. **Nástroje CASE**

V současné době je velice rozšířené používat po objektově orientované analýzy a návrhy IS nástroje zvané CASE (Computer Aided Software Engineering). Tyto nástroje používají diagramy UML a mnoho dalších nástrojů pro lepší zachycení všeho, co je potřeba při návrhu systému. Dnes je již zcela běžné, že při tvorbě středního a většího softwaru jsou použity nástroje CASE. Dražší verze těchto nástrojů dokážou použít UML diagramy k vytvoření základní kostry kódu celého projektu a tím usnadnit programátorům část práce.

Dnes je na trhu mnoho těchto nástrojů, jako například Rational Rose od firmy Rational, Select Component Architect firmy Celect Business Solution a další. Existují i kreslící nástroje, které dokážou zachytit UML, jako je třeba kreslící program Visio od firmy Microsoft, ten však postrádá pokročilejší funkce.

Doby, kdy k analýze a návrhu stačila pouze tužka a papír jsou již pryč a tak jsou nástroje CASE dnes již nepostradatelnou součástí každé firmy, vyvíjející větší software. (Kanisová a Müller, 2006)

V této práci je použit produkt firmy Sparx systems s názvem Enterprise Architect.

3.12.1. **Použití Enterprise Architekt**

Enterprise Architect je modelovací nástroj jazyka UML, vyvíjen australskou firmou Sparx systems. Program je možné instalovat na platformy Windows a Linux. Enterprise Architect nabízí k datu 19. 3. 2012 verzi 9.3. Sparx systems se vývojem tohoto typu produktu zabývá již 12 let. Jeho finanční dostupnost je, vzhledem k dalším rozsáhlým možnostem, velmi dobrá. Aktuální verze obsahuje UML 2.4.1, modely procesní analýzy BPMN, možnost generování kódu v různých programovacích jazycích a mnoho dalších. (Enterprise Architect, 2008 a UML tools for software development and modelling, 2012)

4. Praktická část

V této kapitole bude proveden postupně sběr požadavků, analýza požadavků na informační systém a následný návrh dalšího postupu ve vybraném podniku. Podnik, pro který je vytvořena analýza, se jmenuje Blue Orange a.s.

Společnost se rozhodla pro vlastní řešení z důvodu, že na trhu neexistuje hotový systém, který by nabízel potřebnou variabilitu. V případě, že by systém společnost poptala u externích firem, šlo by o velice drahé řešení s nejistým výsledkem. Z tohoto důvodu se rozhodla zadat projekt vedoucímu pracovníkovi IT úseku, který má zkušenosti s programování a hlavně s obsluhou celého stávajícího řešení. To zaručuje, že vedoucí projektu bude přesně znát potřeby společnosti.

V první části tvorby analýzy je zapotřebí především podrobná analýza pro členské účty, proto je v analýze věnována velká část tomuto úseku.

4.1. Podnik Blue Orange

„Blue Orange je unikátní centrum s přátelským prostředím, poskytující vysoce kvalitní služby z oblasti fitness, wellness, relaxace a zábavy.“ (Blue Orange a.s., 2012)

4.1.1. **Obecné informace**

Název:	BLUE ORANGE a.s.
Adresa:	Tupolevova 676, Praha 9 – Letňany 199 00
IČO:	26198479
Právní forma:	Akciová společnost
Webové stránky:	www.blueorange.cz

4.1.2. **Historie a popis podniku**

Společnost Blue Orange (dále jen společnost) byla poprvé otevřena v dubnu roku 2003 v pražských Letňanech. Jde o unikátní centrum s přátelským prostředím, které poskytuje služby z oblasti wellnes, relaxace, fitnes a zábavy. Středem centra je privátní health club, který nabízí nadstandardní péči o zákazníky.

Centrum dále nabízí čtyřhvězdičkový hotel, vyhlášenou restauraci s letní zahrádkou, beauty salón, konferenční místnosti a pro milovníky golfu chippovací hřiště.

Veškeré služby jsou vyjma health clubu jsou dostupné pro nečleny klubu, pokud někdo chce navštívit health club, musí se nejdříve stát členem.

4.1.3. **Popis jednotlivých oddělení podniku**

- Health club:

  Fitness centrum s nadstandardními a kvalitními službami, jako je speciální péče o klienty s osobní a přátelský přístup s důrazem na dodržení čistoty prostředí.
- Restaurace:

  Vyhlášená restaurace především středomořskými a českými specialitami. V letních dnech je v restauraci otevřena zahrádka a nabídka jídel rozšířena o speciality z grilu. Restaurace dále nabízí business snídaně, týdenní menu, sommelierské večery, home party service, jídlo sebou a různé akce, jako jsou rauty a svatby.
- Beuty salon:

  Zahrnuje nabídku manikúry, pedikúry, kadeřnictví, kosmetické, vizážistické služby a poradenství z těchto odvětví.
- Hotel:

  Centrum nabízí čtyřhvězdičkový hotel s 12 dvoulůžkovými pokoji.
- Konferenční sály:

  Prostory pro pořádání různých školení, porad, firemních schůzek, prezentací či společenských setkání. K dispozici jsou celkem tři sály.

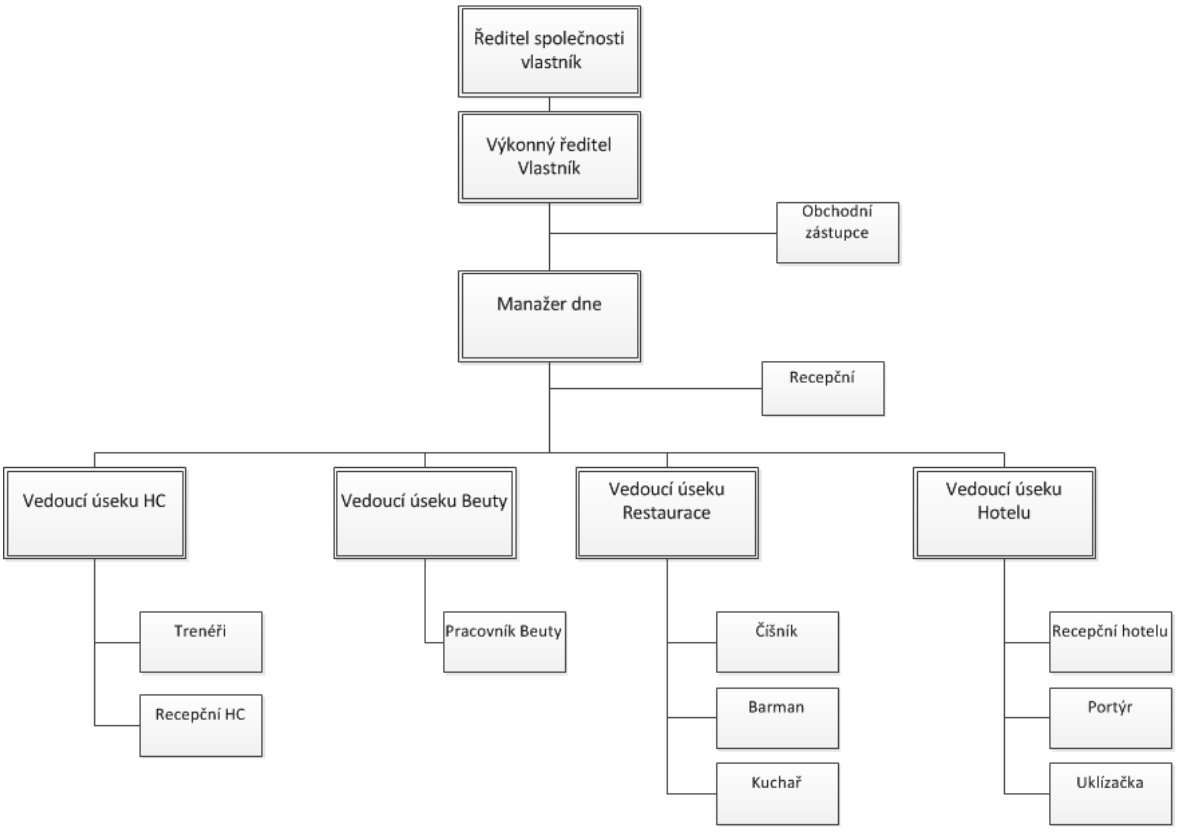
  1. Velký konferenční sál, s kapacitou 60 osob a velikostí 80m^2,
  2. salónek 1, s kapacitou 14 osob a velikostí 28 m^2,
  3. salónek 2, s kapacitou 20 osob a velikostí 33m^2.

Ke všem konferenčním sálům je k dispozici připojení k internetu, telefonní linka, fax, kopírka a hlídané parkoviště.

- Chippovací hřiště:
  Nabízí 3 profesionální greeny s modelovaným povrchem pro trénink patování a pískovou překážku.
- **Ukázková prodejna výrobků značky Boblbe-e:**
  Firma vlastní výhradní zastoupení na prodej značky Boble-e. Tyto výrobky je možné zakoupit na recepci centra, u kterého je i showroom. Společnost také provozuje internetový obchod na internetovém drese www.boblmania.cz.

4.1.4. **Organizační struktura**

Na následujícím obrázku je znázorněna orientační organizační struktura.



Obrázek 4.1 Organizační struktura podniku

4.2. Výběr řešení

Společnost nejdříve uvažovala o zavedení existujícího řešení na místo tvorby vlastního, ale při poptávání existujících aplikací u různých společností zjistila, že nabízený software nepokrývá veškerou požadovanou funkčnost.

Další variantou bylo upravení existujících produktů, které jsou na trhu k dispozici. Bohužel ani tato varianta neprošla přes velké nároky na funkcionalitu, navíc si některé společnosti ani netroufli upravovat své produkty na takto specifické požadavky.

Poslední možnou variantou je vytvoření nového softwaru na zakázku. Firmy, které odpověděli na poptávku, nabídly velmi drahé řešení. Z toho důvodu se vedení společnosti rozhodlo pro vytvoření interního týmu, který s pomocí externistů vytvoří požadovaný software.

4.3. Projektový záměr

Název projektu

Změna informačního systému (zkratka projektu ZIS)

Současný stav:

Společnost Blue Orange má nyní vlastní informační systém PATO@GASTRO od firmy Abaton Praha s.r.o., který propojuje funkčnost celé společnosti. Tento systém je zatím dostačující, ale do budoucna je nutné stávající systém vyměnit.

To je způsobeno skutečností, že firma vytvářející a spravující software ho již nehodlá podporovat, a ani nechce prodat zdrojové kódy. Tím pádem je tento software do budoucna nepoužitelný, protože neumožňuje zavedení jakékoliv změny. Tento problém již v minulosti společnost potkal, když proběhla změna DPH z 19% na 20% a nebylo možné tento údaj změnit nijak jinak, než zásahem programátora. Tím pádem vzniká firmě obrovské riziko, které se může projevit takřka kdykoliv nějakou nepředvídatelnou událostí. Proto se společnost rozhodla vytvořit si software sama s pomocí externích pracovníků.

Cíl projektu

Systém musí propojovat celou organizaci a vytvářet spojení mezi jednotlivými úseky společnosti tak, aby poskytl co možná největší pohodlí a volnost pro klienty. Je tedy zapotřebí vytvořit zcela nový informační systém, který bude obsahovat funkčnost IS

a navíc nové funkce, které již nebylo ?možné nebo efektivní přidávat do starého IS. Systém bude zpracován týmem externistů, které povede vedoucí pracovník IT oddělení společnosti. Aby IS splňoval veškeré požadavky, je zapotřebí vytvoření softwaru na míru z důvodu velké složitosti systému. Toto rozhodnutí vychází z průzkumu trhu vedoucím pracovníkem IT.

Externí pracovníci místo jiné firmy jsou zvoleni ze dvou důvodů.

- Uspoření nákladů při tvorbě zakázkového IS,
- vlastnictví zdrojových kódů a mít vývoj IS pod kontrolou.

Cílové skupiny

Hlavním aktérem v celém systému jsou zaměstnanci společnosti. Vedoucí pracovníci budou potřeba především ve fázi sběru požadavků a ostatní zaměstnanci budou potřeba při testování aplikace v testovacím provozu.

Zdůvodnění potřebnosti projektu

Největším přínosem bude vlastnictví zdrojových kódů a možnost tedy kdykoliv jakéhokoliv zásahu do systému bez ohledu na jinou firmu.

Přínos pro cílové skupiny

Systém bude v mnoha odhledech stejný jako stávající, ale budou přidány funkčnosti, které zaměstnanci požadovali a budou vyhodnoceny jako přínosné ve fázi sběru požadavků.

Klíčové aktivity projektu

- Schválení projektováno záměru,
- vymezení rozpočtu na projekt,
- sběr požadavků na systém,
- tvorba analýzy systému,
- návrh systému,
- programování systému,
- testování systému,
- migrace dat,
- zkušební provoz,
- ostrý provoz.

Harmonogram projektu

T – zahájení projektu

Harmonogram začíná schválením projektového záměru, v projektu jsou vždy uvedeny časy, kdy by každá fáze měla končit. Fáze na sebe nemusí navazovat a můžou probíhat najednou. Odhadovaná doba zavedení softwaru je 6 měsíců.

Činnost	dnů
schválení projektováno záměru	T+5
vymezení rozpočtu na projekt	T+10
sběr požadavků na systém	T+20
tvorba analýzy systému	T+40
návrh systému	T+50
příprava prostředí	T+80
programování systému	T+100
testování systému	T+120
validace a migrace dat	T+140
zkušení provoz	T+179
spuštění ostrého provozu	T+180

Tabulka 4.1: Harmonogram projektu

Rizika projektu

- Projekt nebude vytvořen v požadovaném termínu,
- projekt je finančně podhodnocen,
- projekt se nepodaří dokončit.

Realizační tým

Popis funkce	Jméno
vedení projektu, koordinace pracovníků, nasazení a testování systému	Jakub Frieb
analytik, dokumentarista	Michal Voják
Programátor / architekt softwaru	Ondra Krátký
Grafik	Martin Henych

Tabulka 4.2: Popis rolí v realizačním týmu

Rozpočet

Předběžný rozpočet na projekt je stanoven rozpočet 500 000Kč bez DPH.

Udržitelnost projektu

Projekt bude nadále ve vlastní správě společnosti. Správu bude zastávat oddělení IT. Bude smluvně dohodnutá součinnost programujícího programátora na 5 let. Při programování softwaru bude kladen velký důraz na správném okomentování kódu, aby bylo možné najmout v případě nedostupnosti programátora, který software dodá, možnost úpravu zadat někomu jinému. Správné okomentování kódu bude hlídáno vedoucím pracovník IT.

4.4. Sběr požadavky

V této kapitole se nachází zápis požadavků převážně formou případů použití. Požadavky na nový systém, vycházejí ze schůzek se zaměstnanci, kteří se systémem pracují. Šlo hlavně o vedoucího pracovníka IT a vedoucí jednotlivých úseků. Dále byl zkoumán stávající systém pro vytvoření představy o jeho kompletní funkčnosti.

4.4.1. Jak číst dokument s požadavky a pro koho je určen

První část dokumentu nepředpokládá velké znalosti analýzy IS. V druhé části jsou použity nástroje jazyka UML a to případy použití (use case) na které je zapotřebí buďto znalost tohoto jazyka, nebo analytika znalého tohoto nástroje. Dokument obsahuje slovník pojmů užitých ve sběru požadavků a následné analýze, funkční a nefunkční požadavky kladené na systém a případy použití systému.

Dokument je určen pro zadavatele analýzy a slouží pro vytvoření popisu obecné funkcionality IS v dané společnosti.

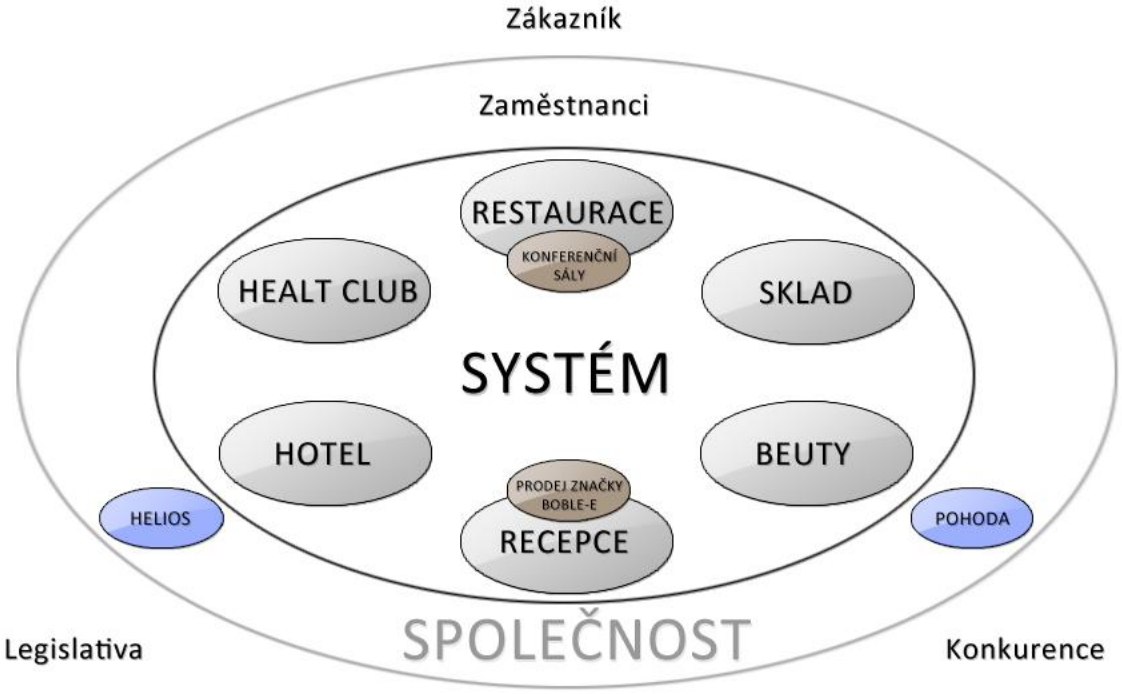
4.4.2. Obecný popis sytému

Společnost se dělí na pět základních úseků, kterými jsou:

- Restaurace,
- health club,
- hotel,
- beauty,
- recepce.

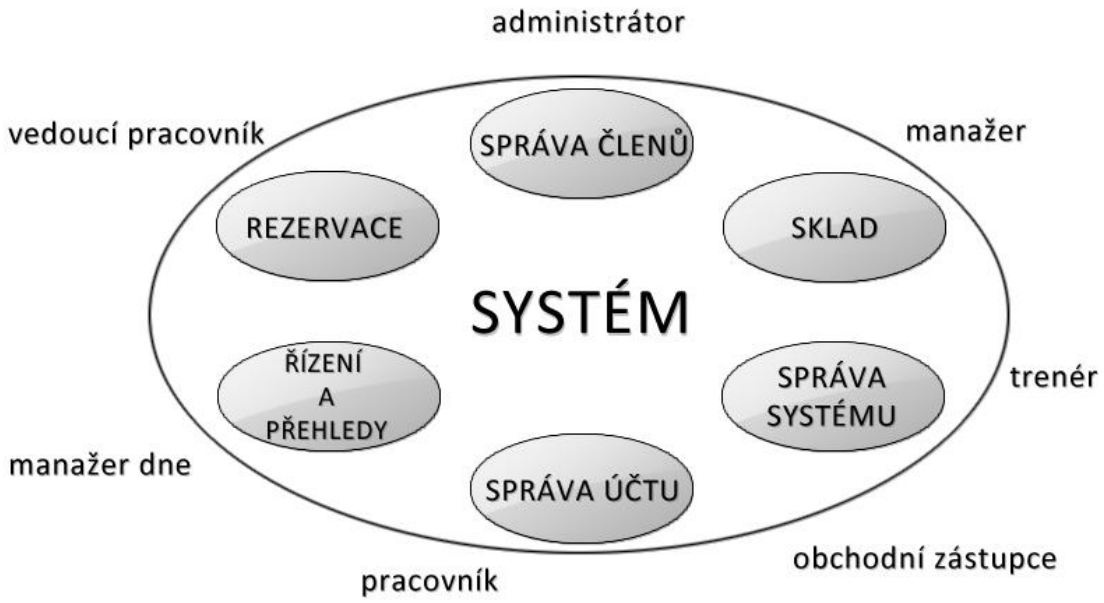
Tyto bloky jsou vzájemně propojeny přes účty klientů a nabízené služby. Každý z těchto úseků má přístup do skladu, kde jsou umístěny suroviny a produkty. Obrázek 4.2

znázorňuje zjednodušený nákres, jak systém propojuje jednotlivé úseky ve společnosti, jaké je okolí systému a následně celé společnosti.



Obrázek 4.2: Zjednodušené znázornění systému ve společnosti.

Naopak na obrázku Obrázek 4.3 je možné vidět zjednodušené analytické zakreslení systému a jeho funkčního propojení.



Obrázek 4.3: Funkční propojení systému

Firma nevlastní procesní modely a ani neuvažuje o jejich vytvoření, celá analýza bude tedy vytvářena pouze na základě sběru požadavků.

4.4.3. Slovník projektu

Slovník pojmů definuje veškeré důležité pojmy v projektu. Celý slovník je možné nalézt v příloze číslo 1.

4.4.4. Funkční požadavky

Funkční požadavky budou pro zpřehlednění rozděleny do jednotlivých úseků, které jsou shodné s úseky podniku. Při zápisu požadavků je bráno v potaz základní dědění pravomocí rolí v organizaci. Tedy:

- **Prodejní úsek:**
  Co může dělat pracovník, je schopen dělat i vedoucí daného úseku a potažmo i manažer dne.
- **Dohledový úsek:**

  Co může dělat trenér, je schopen dělat obchodní zástupce, a co může dělat obchodní zástupce, je schopen dělat i manažer.

Více o této problematice dědění rolí viz kapitola 4.4.6 níže.

ID	uživatel	znění	priorita	typ
FUN_1	systém	Systém bude sledovat a hlásit konec členství klientů.	Možný	členové
FUN_2	vedoucí úseku	Systém bude umět přiřadit ke členům garanta.	Možný	členové
FUN_3	manažer dne	Systém bude moct zakládat, upravovat a mazat členské účty.	Nezbytný	členové
FUN_4	manažer dne	Systém bude umět prodlužovat členská konta.	Nezbytný	členové
FUN_5	manažer dne	Systém bude zaznamenávat platby za členské příspěvky.	Nezbytný	členové
FUN_6	pracovník	Systém bude umožňovat pohlížet data členů.	Nezbytný	členové
FUN_7	pracovník	Systém bude umožňovat zapisovat poznámky ke členům.	Nezbytný	členové

ID	uživatel	znění	priorita	typ
FUN_8	manažer dne	Systém bude umět vytvořit záznam o členské kartě.	Nezbytný	členové
FUN_9	manažer dne	Systém bude umožňovat více členských karet.	Nezbytný	členové

Tabulka 4.3 ukazuje funkční požadavky na funkcionalitu členů v systému. Kompletní výčet požadavků je možné najít v příloze číslo 7.2.

ID	uživatel	znění	priorita	typ
FUN_1	systém	Systém bude sledovat a hlásit konec členství klientů.	Možný	členové
FUN_2	vedoucí úseku	Systém bude umět přiřadit ke členům garanta.	Možný	členové
FUN_3	manažer dne	Systém bude moct zakládat, upravovat a mazat členské účty.	Nezbytný	členové
FUN_4	manažer dne	Systém bude umět prodlužovat členská konta.	Nezbytný	členové
FUN_5	manažer dne	Systém bude zaznamenávat platby za členské příspěvky.	Nezbytný	členové
FUN_6	pracovník	Systém bude umožňovat pohlížet data členů.	Nezbytný	členové
FUN_7	pracovník	Systém bude umožňovat zapisovat poznámky ke členům.	Nezbytný	členové
FUN_8	manažer dne	Systém bude umět vytvořit záznam o členské kartě.	Nezbytný	členové
FUN_9	manažer dne	Systém bude umožňovat více členských karet.	Nezbytný	členové

Tabulka 4.3: Funkční požadavky na členy

4.4.5. **Nefunkční požadavky**

Nefunkční požadavky viz Tabulka 4.4, jsou pro zpřehlednění rozděleny do skupin dle jejich zaměření.

ID	znění	priorita
Hardware:		
NEF_1	Systém bude provozován na serveru, ke kterému se budou připojovat jednotlivé stanice,	Nezbytný
NEF_2	Operační systém (dále jen OS) serveru bude Linux (na jeho mutaci nezáleží),	Nezbytný
NEF_3	Systém bude provozován jak na operačním systému Windows 2000 a vyšší, nebo na OS Linux (na jeho mutaci nezáleží).	Nezbytný
Ovládání		
NEF_4	Veškeré ovládání systému z pohledu uživatelů (zaměstnanců společnosti) musí být možné ovládat dotykově přes display.	Nezbytný
NEF_5	Ovládání musí zůstat více méně stejné a to včetně grafického	Možný

ID	znění	priorita
	interface.	
NEF_6	Ať se zaměstnanec registruje na kterémkoliv počítači, vždy má k dispozici pouze práva jemu přidělená. Nezáleží tedy, zda je přihlášen z restaurace, nebo z recepce.	Nezbytný
NEF_7	Systém bude možné spravovat pouze za pomocí jedné aplikace.	Možný
Technologie:		
NEF_8	Systém bude naprogramován v jazyce JAVA s využitím databázové aplikace PostgreSQL.	Možný
NEF_9	Zdrojový kód bude detailně okomentován.	Nezbytný
NEF_10	Databáze musí pojmout 5 000 klientů.	Nezbytný
NEF_11	Databáze musí pojmout 500 000 účtů.	Nezbytný
Dostupnost:		
NEF_12	Systém musí bude možné nainstalovat na jakýkoliv počítač ve firmě, který bude splňovat nároky na OS a bude připojený do interní sítě.	Nezbytný
NEF_13	Odezva systému na pokladnách nesmí být vyšší než 0.25s.	Nezbytný
NEF_14	Systém nebude napojen na jiný systém.	Možný
Zabezpečení:		
NEF_15	Do systému nebude možno přistupovat z jiné než firemní sítě.	Nezbytný
NEF_16	Pro každý přístup bude nutné se přihlásit přes uživatelské heslo nebo magnetický klíč.	Nezbytný
NEF_17	Každý nový počítač se musí zapsat do systému.	Nezbytný
NEF_18	Systém bude provádět záznamy o pohybu zaměstnanců v aplikaci.	Možný
NEF_19	Každý den v určený čas se provede záloha systému na externí zařízení.	Nezbytný

Tabulka 4.4: Nefunkční požadavky

Tyto požadavky zachycují obecnou funkčnost systému, při navrhování struktury jednotlivých tříd je použit průzkum stávajícího systému a jeho databáze.

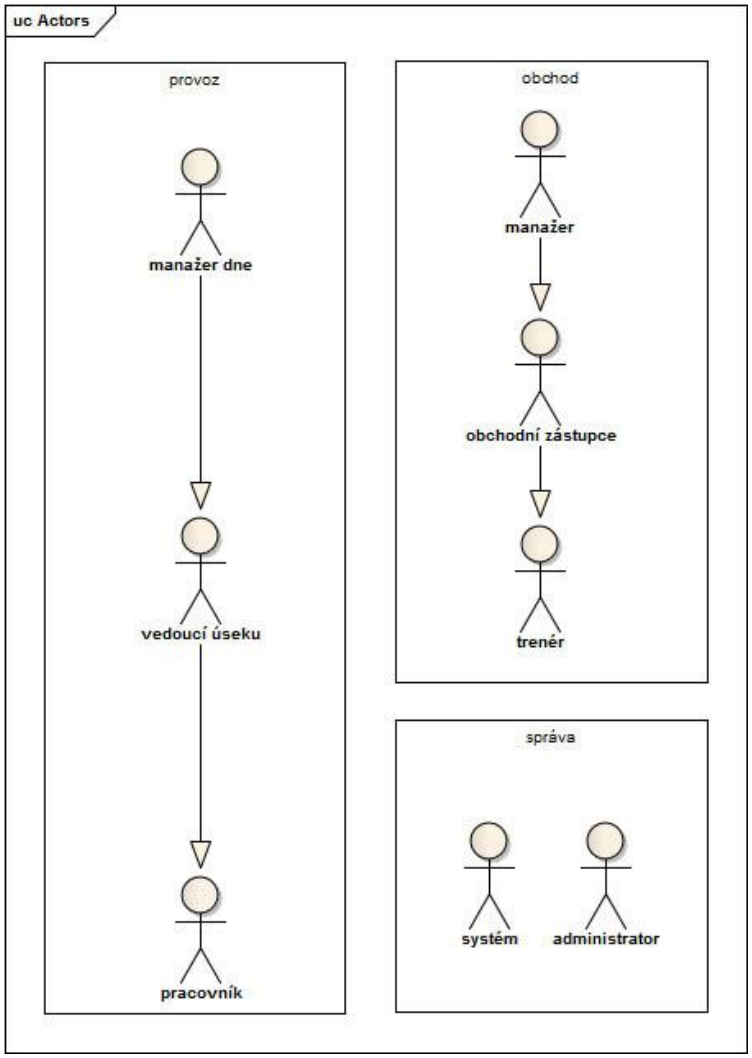
4.4.6. **Identifikace rolí v systému**

Z požadavků vyplývá, že není potřeba speciálních rolí pro každého pracovníka úseku, ale stačí, aby pracovníci, kteří pracují na nižším stupni, měli pouze jednu společnou roli, která je upravována pouze přes práva. Obrázek 4.4 znázorňuje, jakým způsobem probíhá v systému dědění oprávnění k používání určitých částí programu. Z hlediska uživatelů jsou vytvořeny tři základní skupiny uživatelů. První skupina jsou uživatelé, kteří se starají o chod společnosti. Tato skupina se jmenuje provoz a zahrnuje roli pracovník, což je role s nejnižšími právy ve skupině. Další rolí je vedoucí jednotlivých úseků, který disponuje

veškerými právy role pracovník a navíc vlastními právy. Poslední role ve skupině provoz je manažer dne, který vlastní veškerá práva pracovníka a vedoucího dne.

Druhou skupinou je skupina obchodu, která se stará o správu členů a chod firmy. Nejníže postavená je v této skupině role trenéra, který může pouze prohlížet data členů a psát k nim poznámky. Rozšiřující rolí je zde obchodní zástupce, který má na starosti funkce spojené s péčí o členy. Nejvíce postavenou rolí je zde manažer, který má navíc ještě rozšíření sledování veškerého dění z oblasti sledování financí.

Poslední skupina se stará o chod aplikace. V této skupině se nachází administrátor aplikace a role systému, která zde znázorňuje automatizované postupy v aplikaci, jako jsou zálohy systému, různé hlášení apod.



Obrázek 4.4 Znázornění dědičnosti oprávnění v systému

Podrobný popis náplně práce jednotlivých rolí ve společnosti je možné najít v příloze číslo 7.1, Slovník pojmů.

4.4.7. Diagramy případy použití za jednotlivé úseky

V této podkapitole jsou zakresleny všechny případy použití, anglicky use case (dále jen UC), které byly identifikovány z požadavků. Diagramy případů užití jsou rozděleny do jednotlivých bloků, viz předešlý Obrázek 4.3.

Případy užití jsou hlavním výstupem sběru požadavků. Tyto diagramy se dále používají v analýze a následném návrhu.

Správa účtu

Jedním za základních pilířů celého systému je správa účtu zákazníků. O správu účtů se stará především pracovník daného úseku. Na obrázkuObrázek 4.5 je zobrazen diagram, na kterém je vidět, co vše může s účtem v systému pracovník (potažmo i role dědící jeho přístupy) dělat.

Dá se říci, že pracovník je hlavní role, která pracuje s účty v systému. Stará se o jejich vytvoření, přidávání položek a zaplacení.

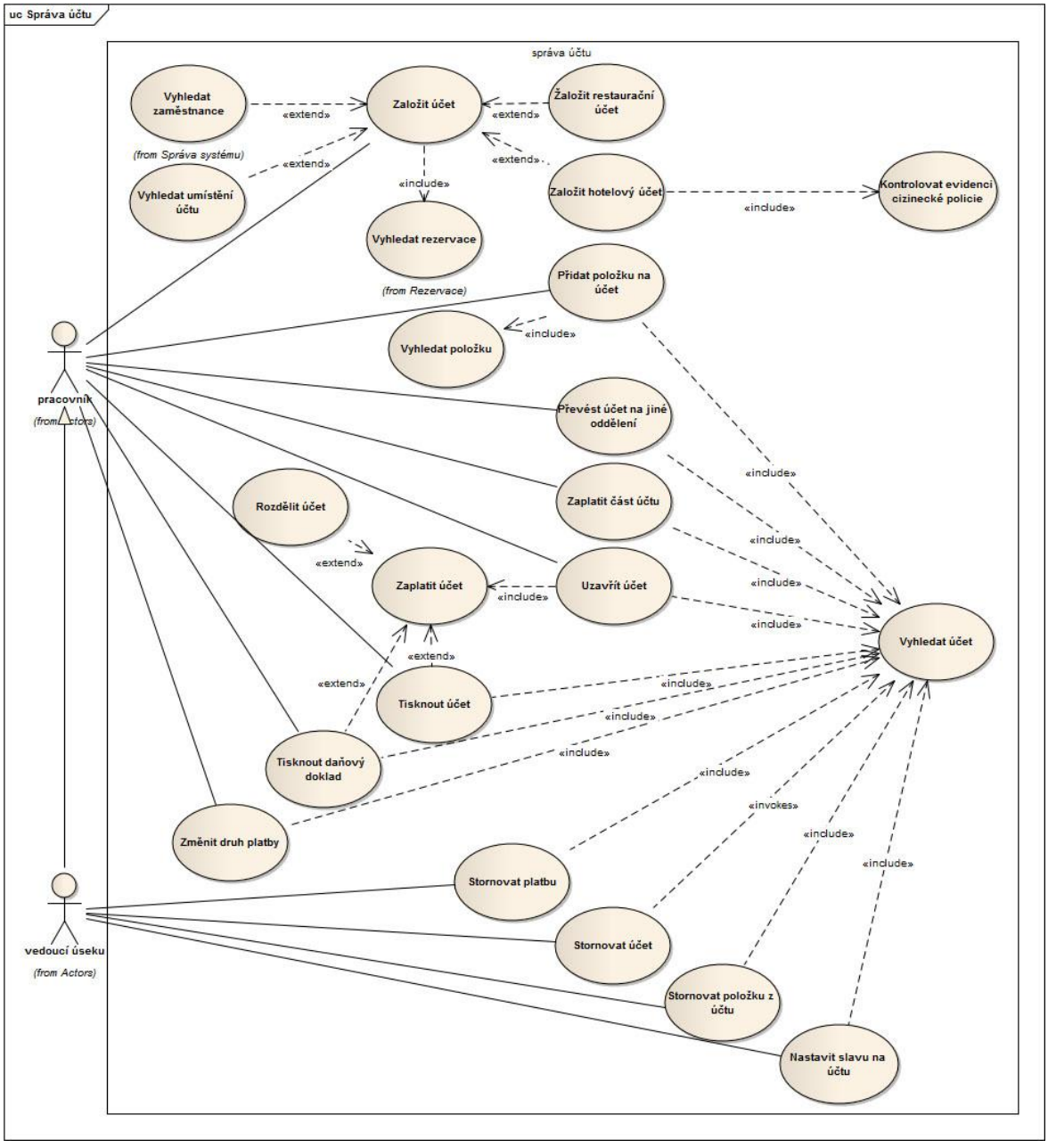
Při založení účtu se určí, zda se jedná o klasický účet, nebo jeho rozšířenou formu, restaurační nebo hotelový účet. U hotelového účtu navíc probíhá kontrola vložených dat v evidenci cizinecké policie. Účet je také možné přiřadit k jednotlivým zaměstnancům, to se využívá tehdy, pokud si chce zaměstnanec něco koupit.

Navíc se při vytváření účtu ověří, zda místo, na které se účet zapisuje, nemá na sobě vytvořenou rezervaci.

Hlavní prací s účtem je vkládání položek na účet. Jak je znázorněno na diagramu, je k této operaci nutné vždy vyhledat účet v systému a vyhledat položky, kterou je potřeba zanést k účtu.

Nakonec je potřeba každý účet zaplatit. K této činnosti se v diagramu váže hned několik UC jako je rozdělení plateb, zaplacení části účtu, tisknutí účtů atd.

Druhou rolí, která se zapojuje do správy účtu je vedoucí daného úseku. Ten má nestarosti především řešení vzniklých problémů, jako jsou storna plateb nebo účtů (UC: stornovat platbu, stornovat účet, stornovat položku z účtu) a zařizování slev (UC: nastavit slevu).

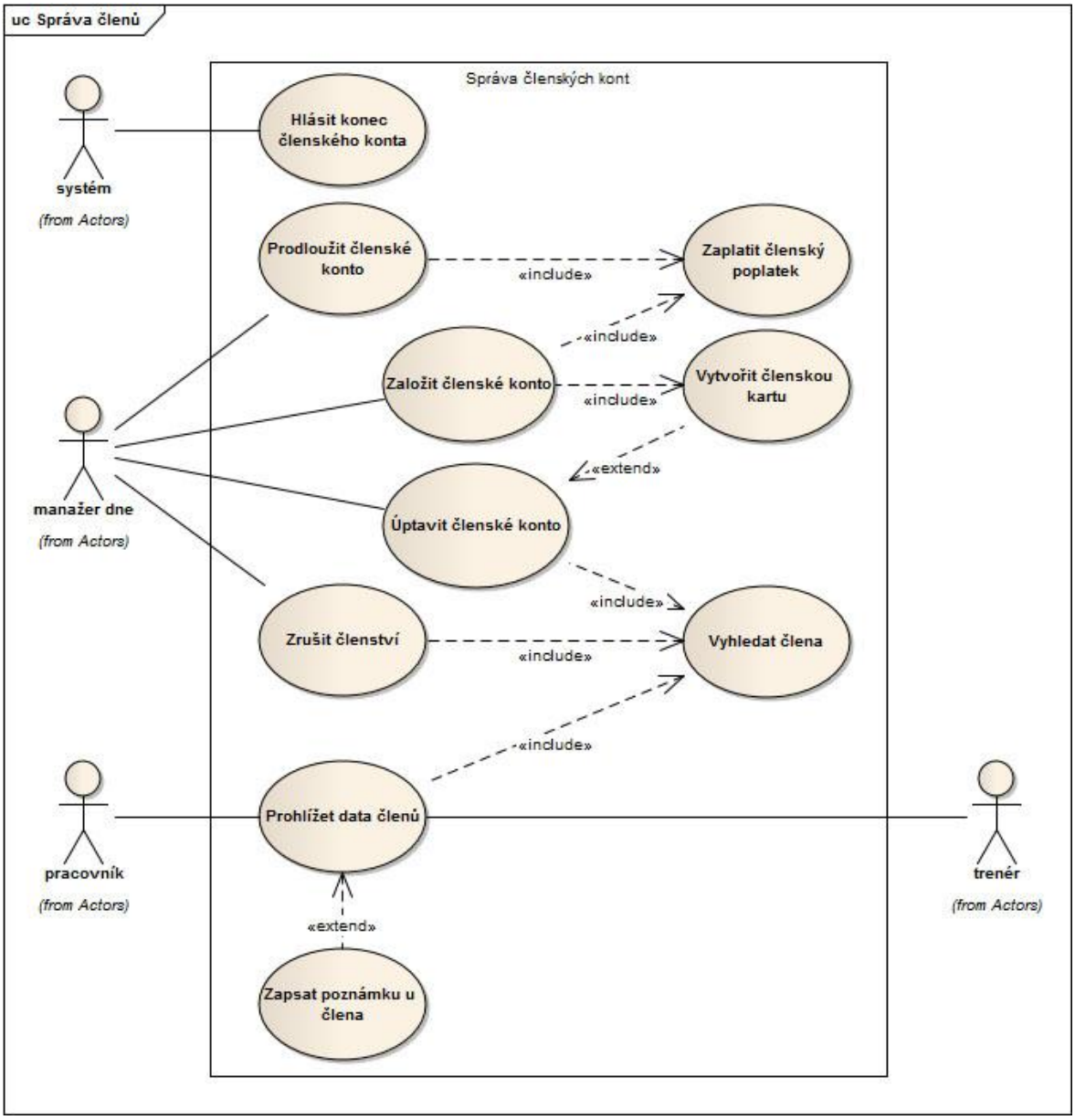


Obrázek 4.5: Diagram znázorňující správu účtu

Správa členů

Druhý diagram na obrázku Obrázek 4.6: Diagram případů použití pro správu členůObrázek 4.6, znázorňuje případ použití pro správu členů. Jak je patrné z UC prohlížet data členů, tak každá role s výjimkou administrátora je schopna nahlížet do dat klientů a vpisovat poznámky k prohlíženým členům (je to způsobeno dědičností mezi rolemi viz Obrázek 4.4).

Hlavní rolí je zde manažer dne, který se stará o celý průběh členských kont. Členská konta vytváří, spravuje i maže. Nedílnou součástí je prodlužování členských kont, které provádí přes UC prodloužit členské konto. V sledování konců členských kont mu pomáhá systém, který hlásí každé konto, které bude za stanovený čas končit.



Obrázek 4.6: Diagram případů použití pro správu členů

Jak bylo zmíněno na začátku kapitoly, vždy pro členský úsek bude rozepsán detailnější zápis.

Scénáře pro UC správy členů:

Následující dvě tabulky znázorňují scénáře pro případy užití členských kont.

Případ použití: Hlásit konec členského konta
ID: UC_29
Stručný popis: Systém zjistí konto, kterému v blízké době vyprší platnost.
Hlavní aktéři: Systém
Vedlejší aktéři: Manažer dne
Vstupní podmínky: 1. Systém je spuštěn. 2. Manažer dne je přihlášen
Hlavní scénář: 1. Případ použití začíná, jakmile existuje účet s blízkou exspirací. 2. Systém zašle zprávu manažerovi dne
Výstupní podmínky: 1. Systém zaslal zprávu.
Alternativní scénář: Žádné.

Tabulka 4.5 Scénář pro případ užití hlásit konec členského konta

Případ použití: Prodloužit členské konto
ID: UC_30
Stručný popis: Prodlouží platnost členského konta
Hlavní aktéři: Manažer dne
Vedlejší aktéři: Žádný
Vstupní podmínky: 1. Manažer dne je přihlášen 2. Má vyhledaného člena vzi UC_36 3. Je splněn UC_34
Hlavní scénář: 1. Manažer dne klikne na tlačítko prodloužit účet. 2. Systém nabídne volbu doby prodloužení. 3. Manažer dne zvolí dobu. 4. Manažer dne zapíše peněžní částku, kterou zákazník platí. 5. Systém napíše, kolik se má vrátit. 6. Manažer dne potvrdí celou transakci. 4. Systém prodlouží planost členského účtu.

Výstupní podmínky: 1. Vytištění potvrzení o prodloužení
Alternativní scénář: **Žádné.**

Tabulka 4.6 Scénář pro případ užití prodloužit členské konto

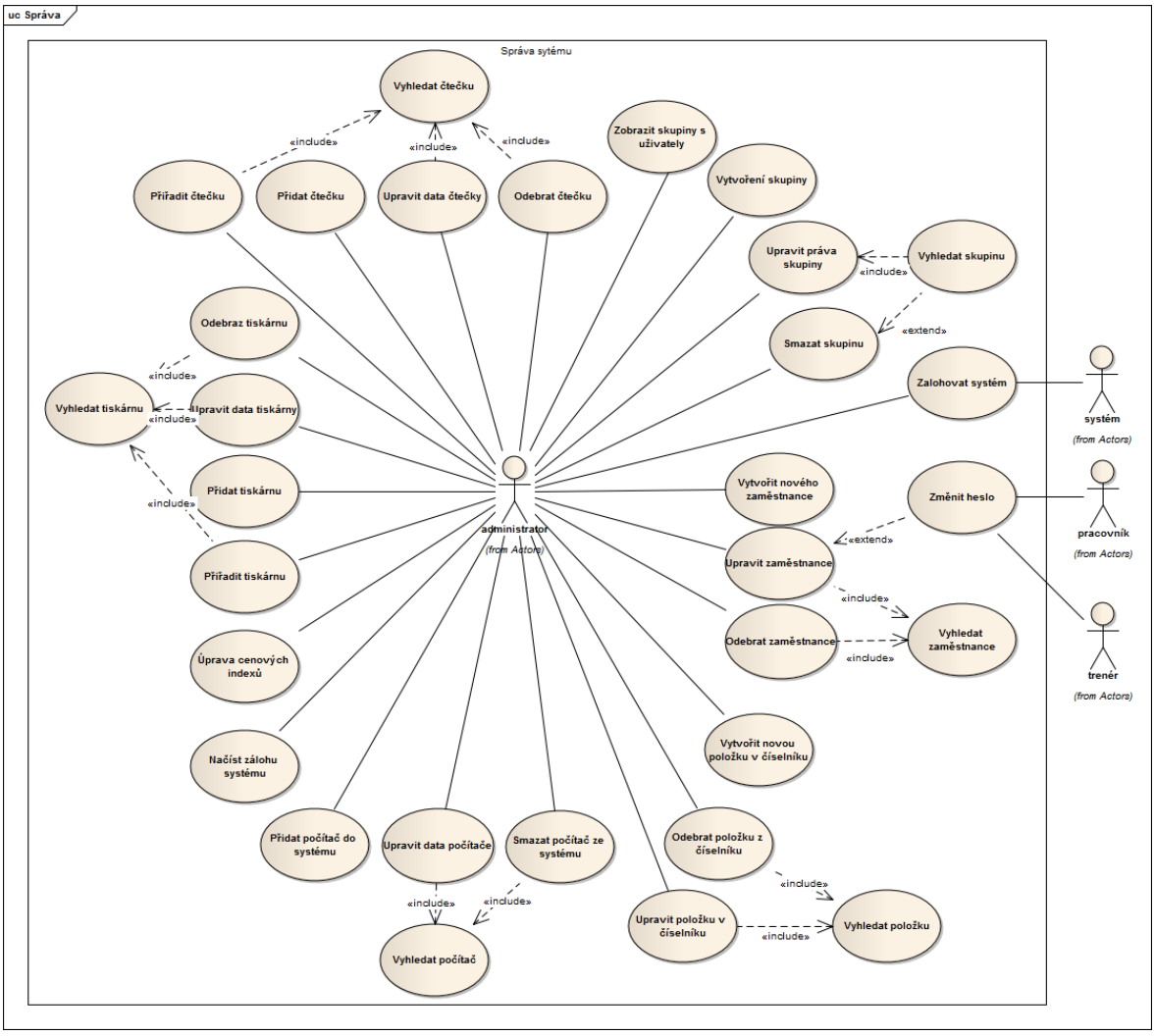
Kompletní scénáře jsou v příloze číslo 7.3.

Správa systému

Na třetím diagramu případů použití viz Obrázek 4.7, je znázorněna správa celého systému. Ústřední rolí je zde administrátor. Pro lepší přehlednost byla role umístěna do středu diagramu, jde však o roli, která stojí vně systému.

Správa systému lze shrnout do správy uživatelský skupin, uživatelů, počítačů, tiskáren, magnetických čteček a číselníků. Všechny UC zobrazují takřka vždy to samé. Možnost zobrazit požadovanou skupinu nástrojů, vytvořit nový element, upravovat jeho data anebo jej smazat. Navíc administrátor může upravovat cenové indexy a zálohovat systém nebo ho načítat ze zálohy. Zálohování systému probíhá i automaticky v nastavený čas. O tuto operaci se stará systém sám.

Jedinou věc, kterou mohou řadový uživatelé měnit ve svém účtu, je jejich heslo viz UC změnit heslo.



Obrázek 4.7: Diagram případu použití pro správu systém

Řízení a přehledy

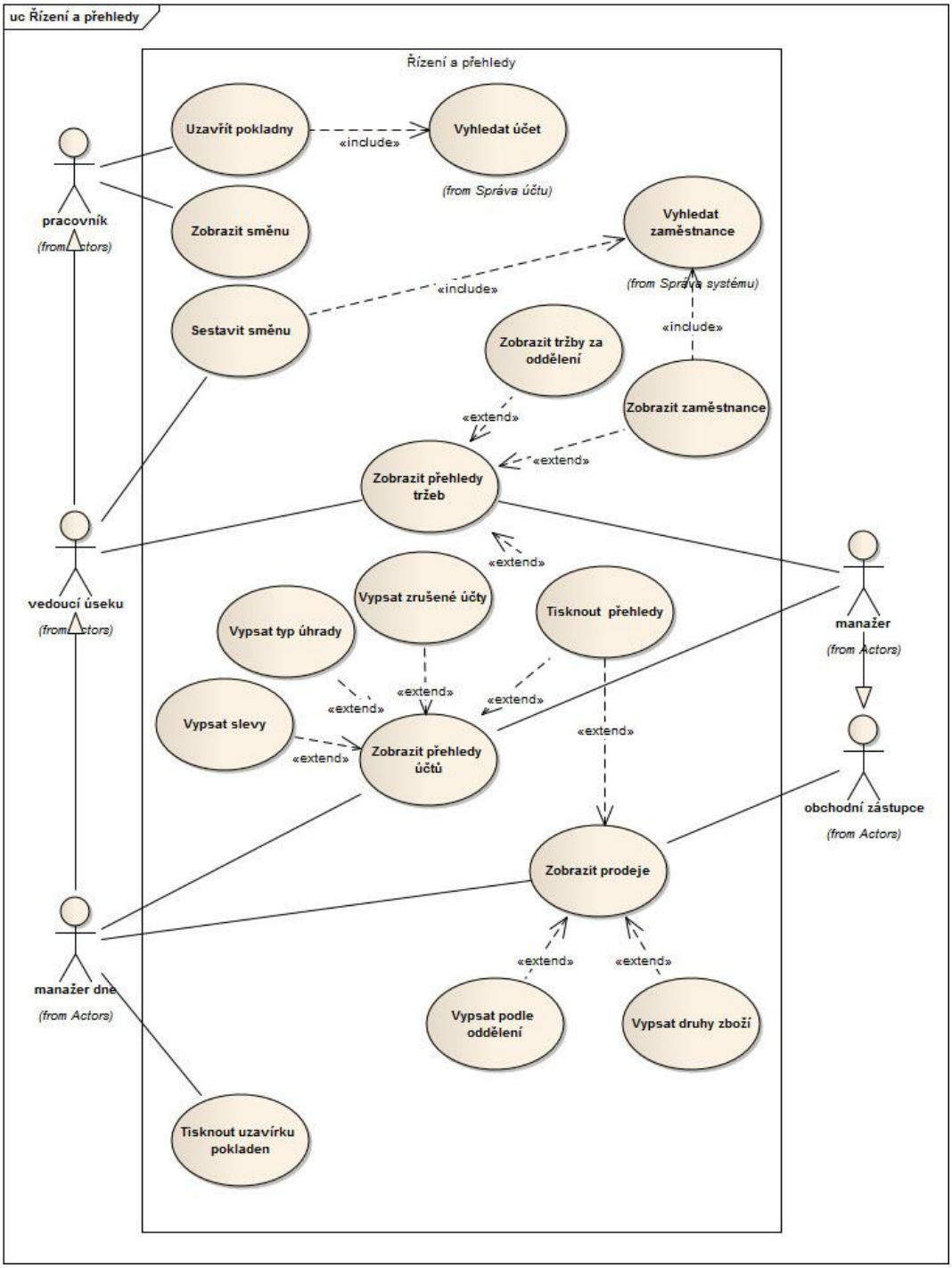
Následující diagram na obrázku Obrázek 4.8, znázorňuje především využití systému pro zobrazení různých finančních přehledů.

Na diagramu je možné vidět, že pracovník uzavírá kasu, viz UC uzavřít kasu. Dále je zde patrné, že pracovník si na terminálu může zobrazit směny, které vedoucí úseku sestavil.

Vedoucí úseku je schopen pomocí UC zobrazit přehledy tržeb kdykoliv zjistit, kolik se ve vybraný čas vyúčtovalo. Je možné zobrazit buďto přehled všech tržeb, nebo setřídit dle oddělení případně za jednotlivé zaměstnance.

Manažer dne má navíc možnost zobrazit přehledy přes všechny účty, které je možné třídit dle slev, typu úhrady nebo si nechat zobrazit pouze zrušené účty. Tuto funkčnost může používat také role manažera. Manažer dne se také stará o tisknutí uzavírek pokladen.

V neposlední řadě je možné vypsat prodeje, které umožňují filtraci dle oddělení nebo druhu zboží. K těmto funkcím má přístup jak manažer dne, obchodní zástupce a manažer.



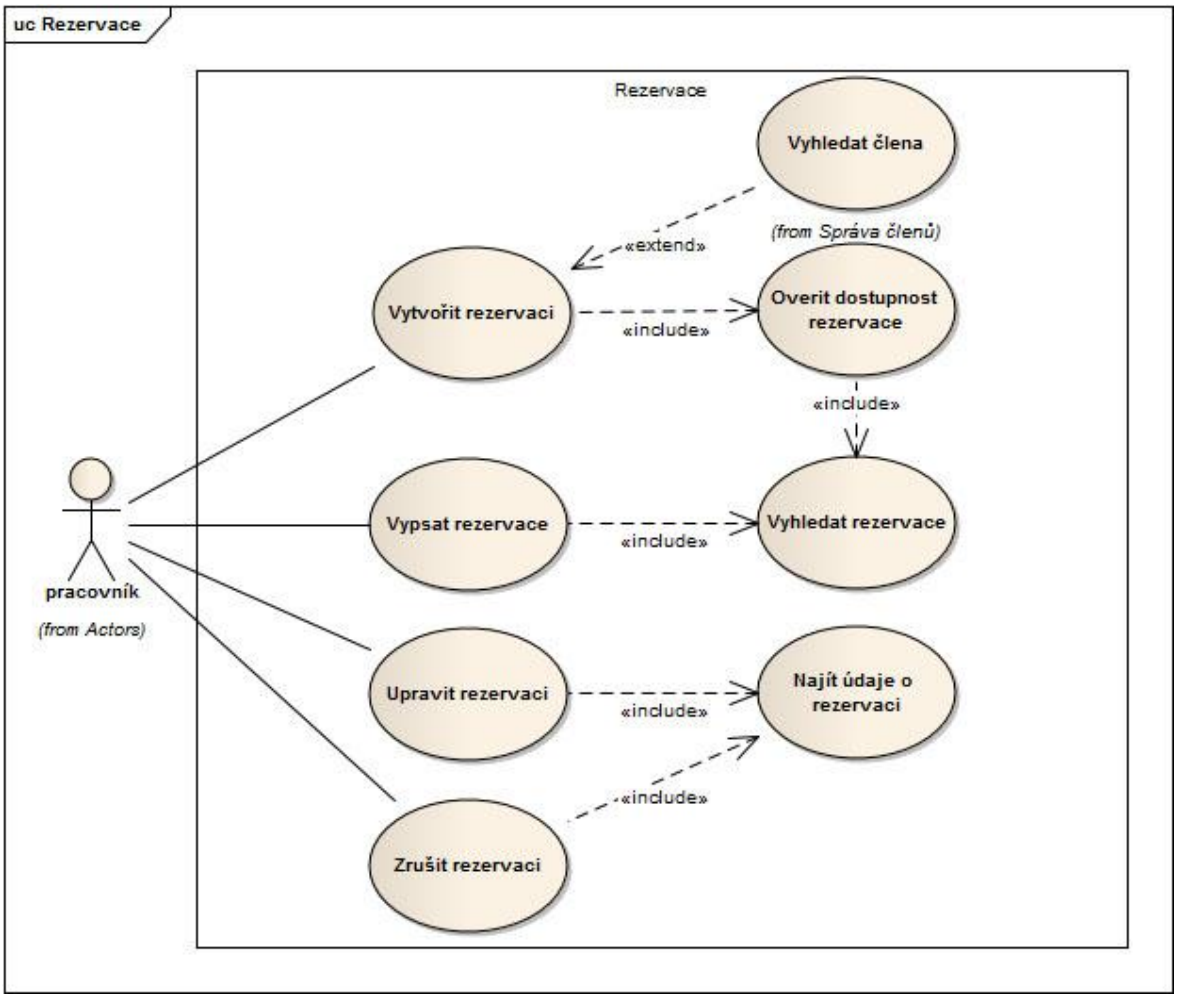
Obrázek 4.8: Diagram případu použití pro řízení a přehledy

Rezervace

Předposledním diagramem znárodněn na obrázku Obrázek 4.9, zobrazuje případ použití pro rezervaci v systému. Hlavním aktérem je pracovník, který se stará o celý proces rezervací.

Při vytváření je vidět, možnost rezervaci přiřadit ke členskému účtu. Při vytváření se zjistí dostupnost vybraného data.

Dále je možné rezervace upravovat, vypisovat nebo rušit.



Obrázek 4.9 Diagram případu použití pro správu rezervací

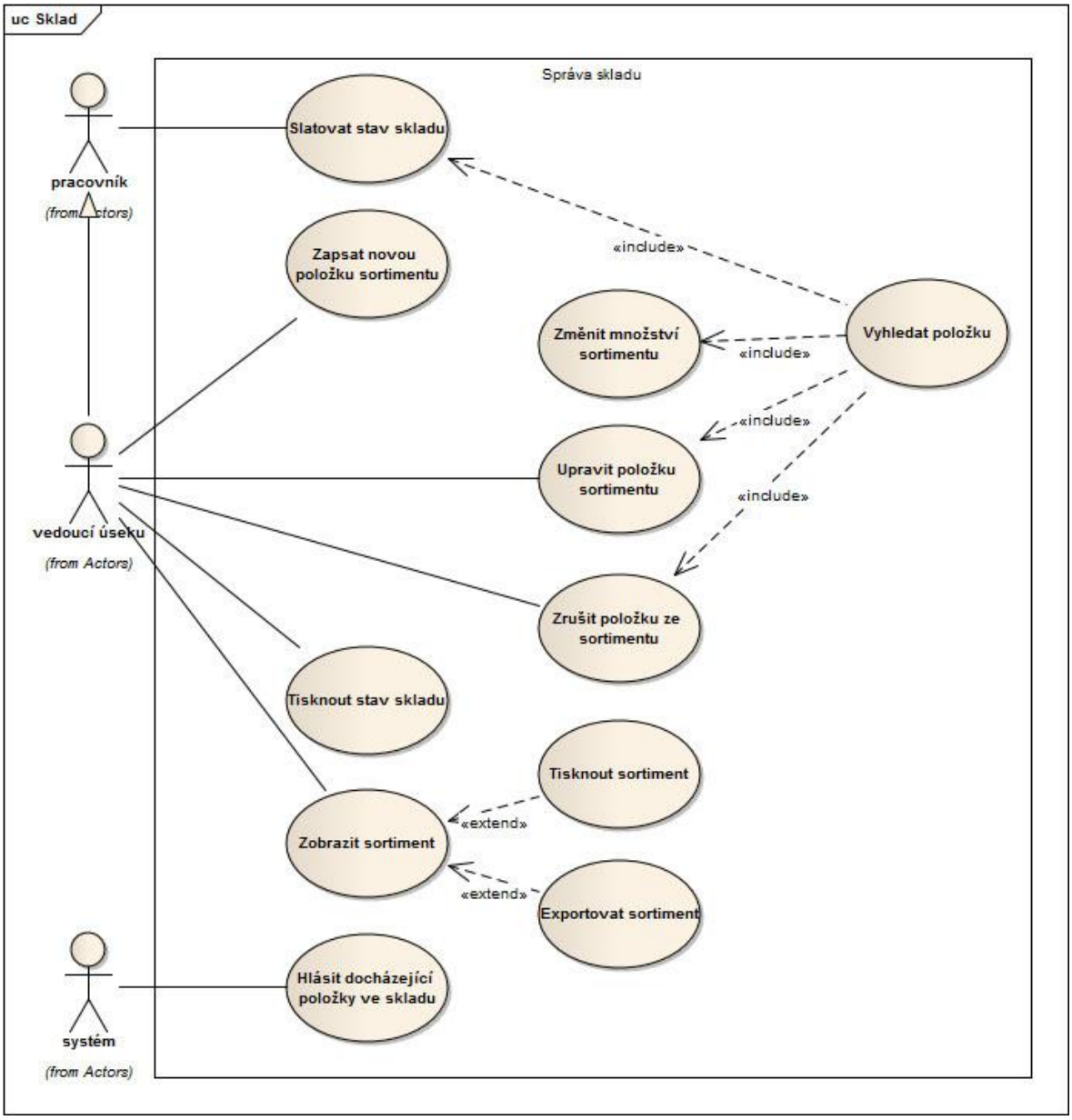
Sklad

Poslední diagram je správa skladu, kterou zobrazuje Obrázek 4.10. Zde je ústřední rolí vedoucí úseku, který spravuje celí sortiment na daném oddělení od zapisování nového sortimentu, doplňování jeho množství, upravování jeho dat, až jeho konečné odstranění.

Celý sortiment má vedoucí úseku možnost zobrazit a následně buďto vytisknout nebo vyexportovat do formátu xls nebo PDF.

Systém zde napomáhá k hlídání docházejícího množství, který zašle hlášení, pokud sortiment klesne k nastavené hranici

Pracovník má možnost pouze sledovat stav nákladného sortimentu.



Obrázek 4.10 Diagram případu použití pro správu skladu

Tímto krokem končí fáze sběru požadavků a začíná fáze analýzy.

4.5. Analýza IS

Tato podkapitola je věnována hledání analytických tříd a jejich interakci s jednotlivými UC.

4.5.1. Analytické třídy

V následující kapitole jsou popsány všechny analytické třídy, které byly identifikovány s nasbíraných požadavků na systém. První a jistě nejkomplikovanější diagram tříd jsou třídy spojené s účtem v podniku.

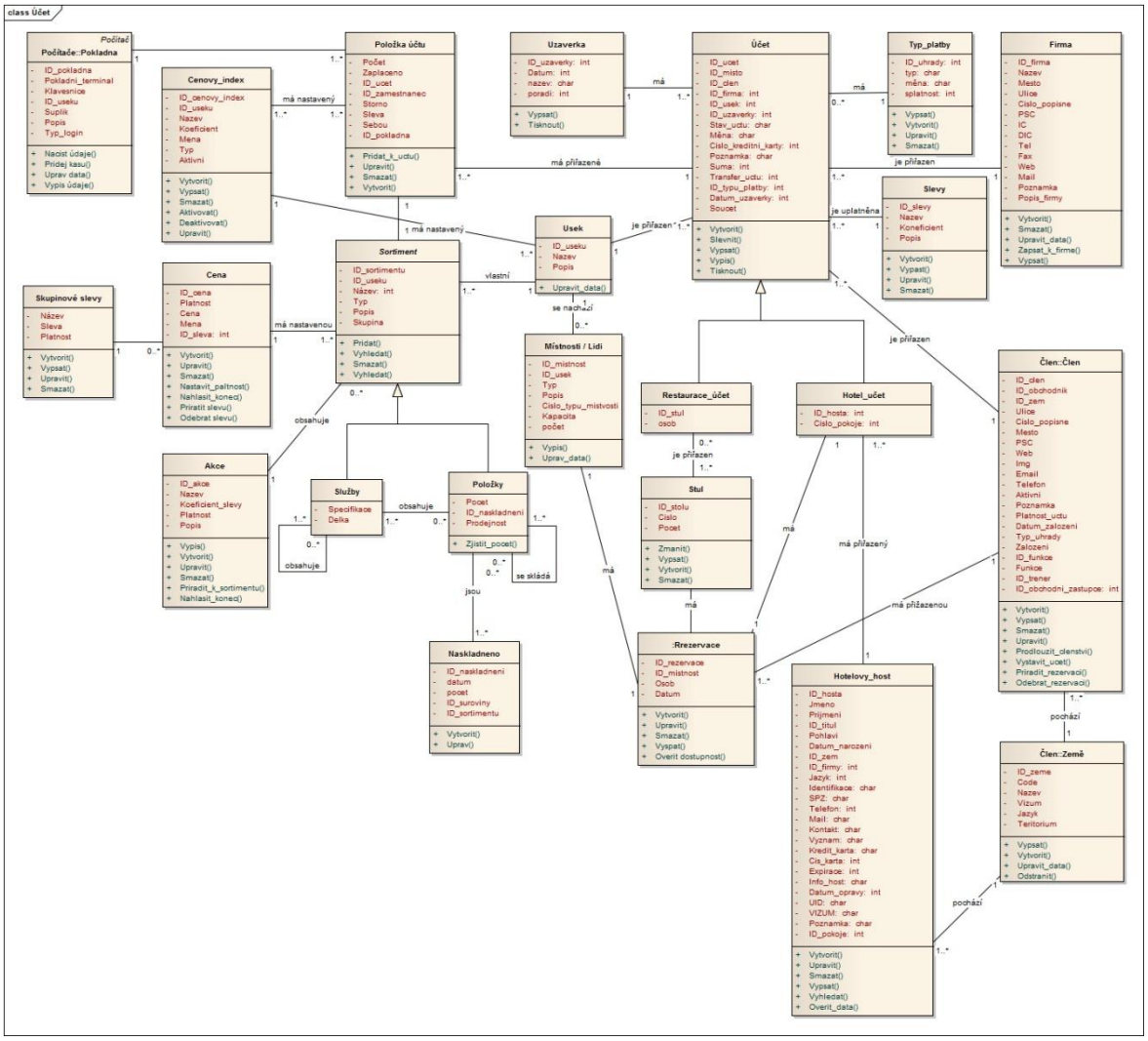
Diagram tříd pro účet

Jde o znázornění tříd v okolí hlavní třídy účet. Tato třída spojuje v zásadě celý systém. Účet může být buďto klasický, nebo rozšířený na hotový či restaurační. Z diagramu na obrázku Obrázek 4.11, je nalevo vidět hlavní propojení na třídu položka účtu. Těchto položek může být ke třídě účet připojeno libovolné množství, minimálně však jedna. Na tyto položky jsou pak připojeny ostatní třídy jako sortiment, jeho cena, cenové indexy a pokladna, na které bylo účtováno. Po bližším zkoumání je zjištěno, že sortiment se dělí na služby a položky (zboží, produkt nebo ingredience).

V levé části od účtu je napojení na jednotlivé úseky společnosti, pro jasnou identifikaci, v jakém úseku se účet nachází.

V pravé části je naopak vidět, rozšíření účtu buďto na firemní nebo na členský. A záznam o platbách a slevách na účtu.

Poslední částí jsou rezervace v systému, které se dají přiřadit buďto k hotelovému účtu, restauračnímu účtu nebo podnikové místnosti.

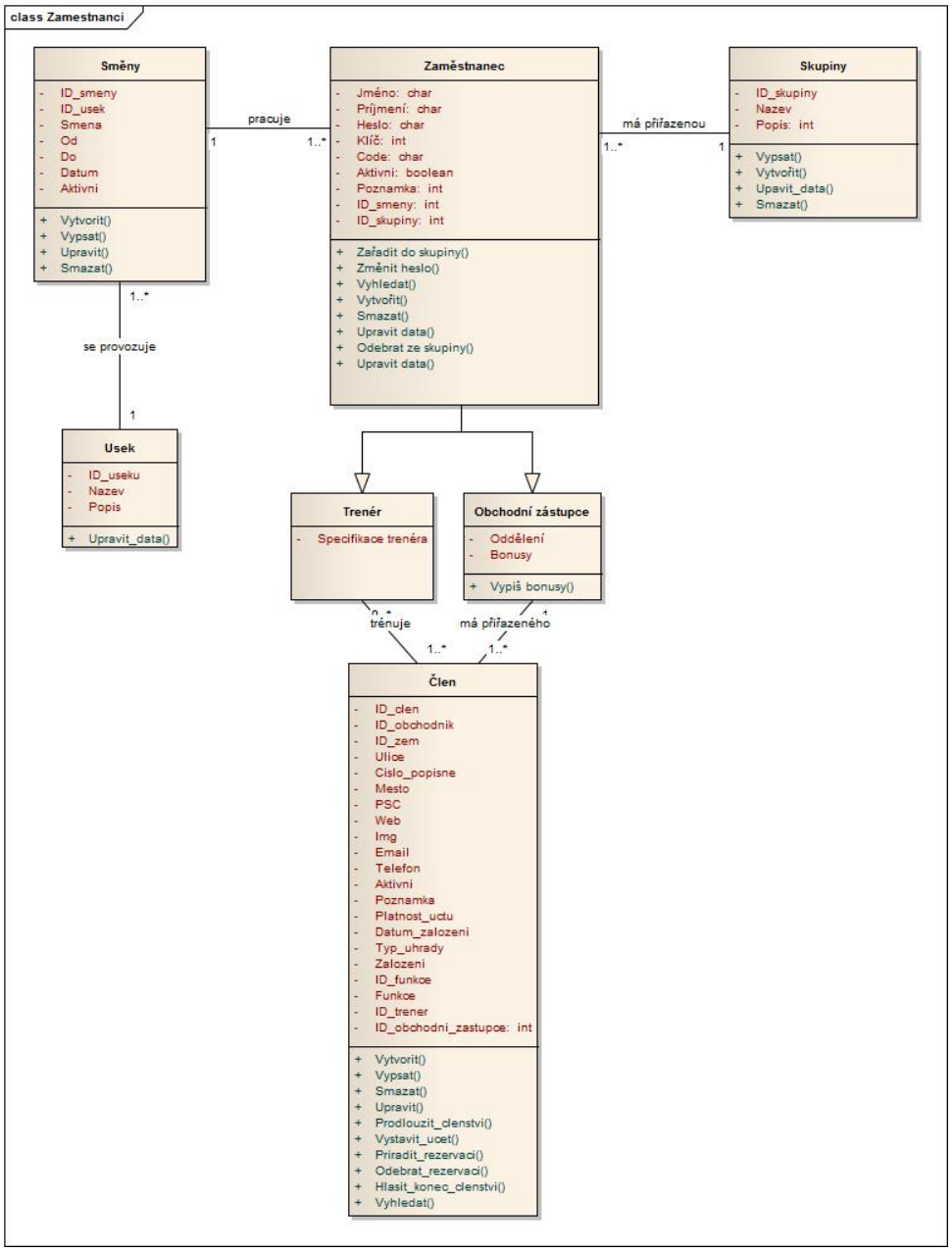


Obrázek 4.11 Diagram tříd pro okolí třídy účet

Diagram tříd pro zaměstnance

O poznání jednodušší diagram tříd znázorňuje zaměstnance ve společnosti. Na obrázku Obrázek 4.12 je vidět, že hlavní třídou je třída zaměstnanec. Na tu se napojují dvě třídy, kterými jsou skupiny, určující do jaké skupiny uživatelů bude zaměstnanec patřit a také třída směn.

Třída zaměstnanci obsahuje ještě dvě specifikace. První jsou trenéři, kteří jsou rozšířeni o atribut specifikace trenéra. Druhou jsou pak obchodní zástupci, kteří jsou rozšířeni o atribut bonusů a zápis o jakou specializaci obchodní zástupce má.

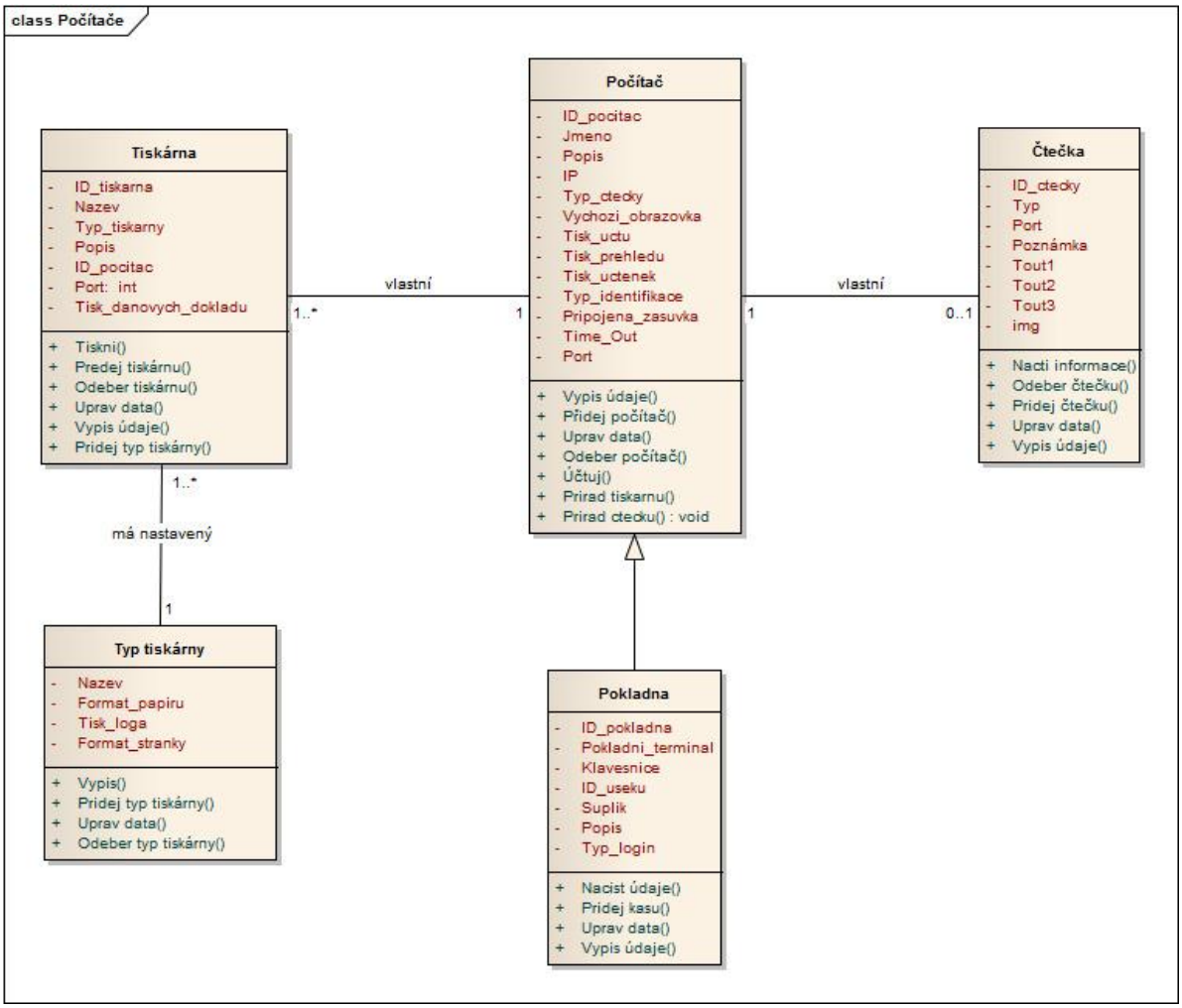


Obrázek 4.12 Diagram tříd pro zaměstnance

Diagram tříd pro počítače

Na následujícím diagramu je znázorněno propojení tříd zastupující počítače a jeho komponenty. Na obrázku Obrázek 4.13, je znázorněno propojení počítačů na tiskárny, jejich typy a čtečky magnetických kódů.

Speciálním případem počítače je pak pokladna, která je rozšířením klasické třídy počítač. Tato třída v reálném světě prezentuje klasickou pokladnu pro placení.



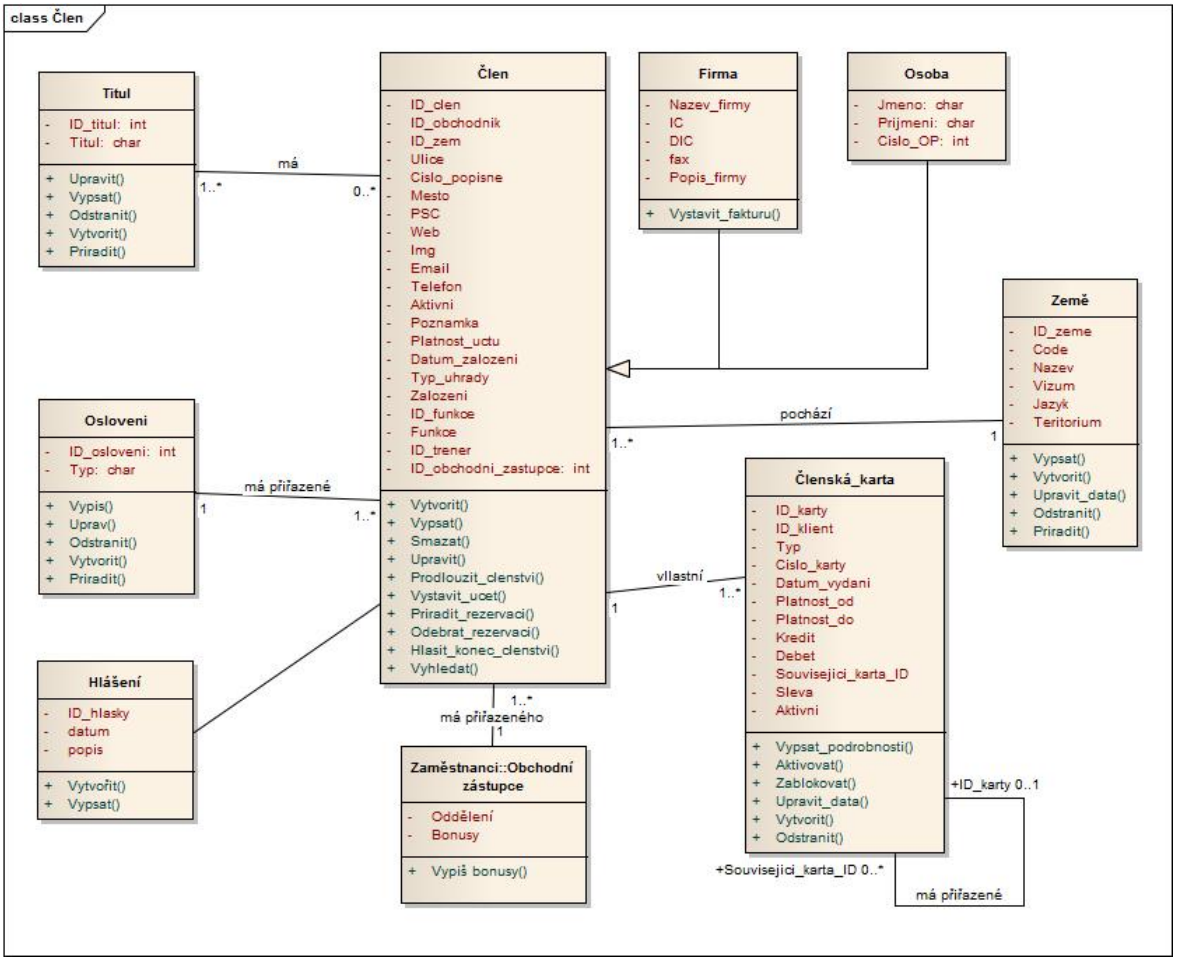
Obrázek 4.13 Diagram tříd pro počítače

Diagram tříd pro členy

Posledním diagramem abstrakcí tříd jsou třídy okolo členských kont. Na obrázku číslo Obrázek 4.14, je zobrazena třída člen, na kterou jsou napojeny dvě rozšiřující třídy a to titul a oslovení[7].

Jako člen může být registrovaná buďto firma nebo osoba a to z jakékoliv země. Každý člen vlastní kartu, na kterou se dají vytvářet další karty.

Poslední propojení je se třídou obchodní zástupce, která znázorňuje navázání každého člena na svého obchodního zástupce, na kterého se mohou členové obracet.

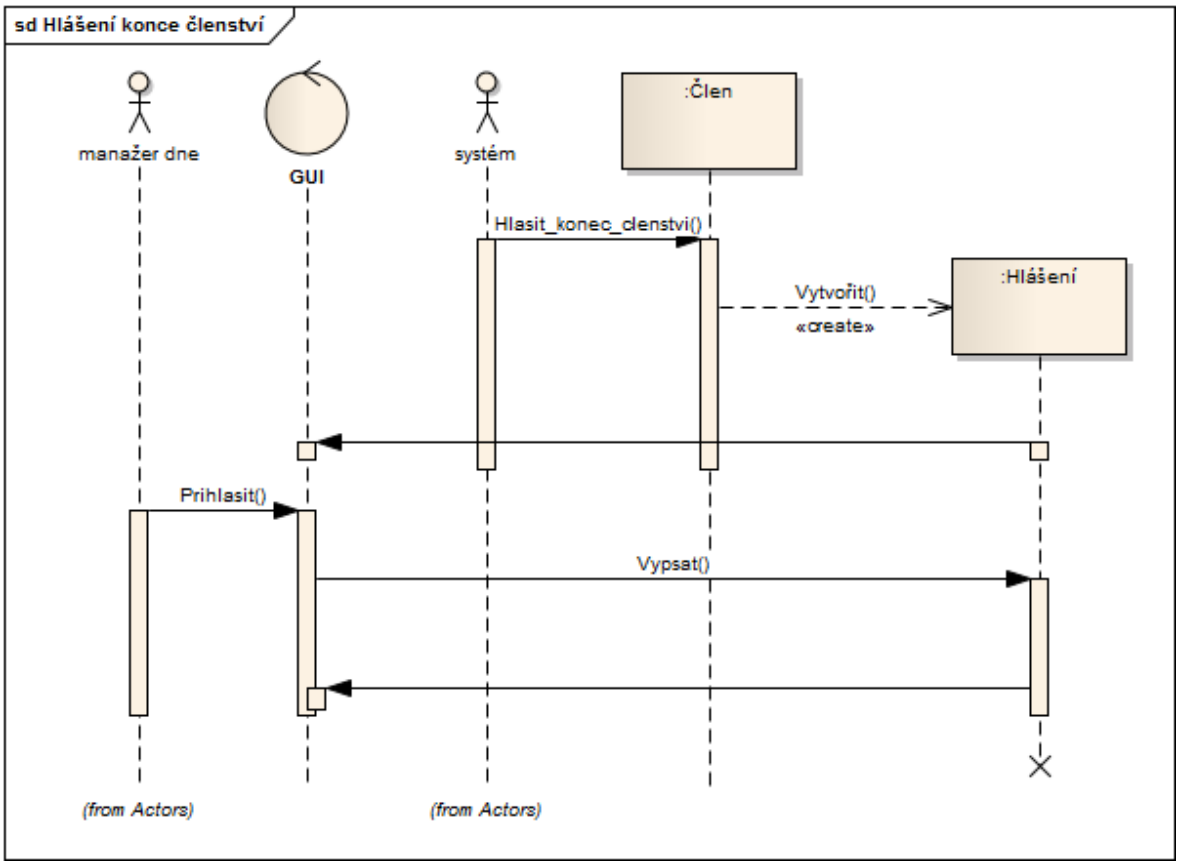


Obrázek 4.14 Diagram tříd pro členy

[7] sloučí pro určení jakým způsobem klienta oslovit při příchodu

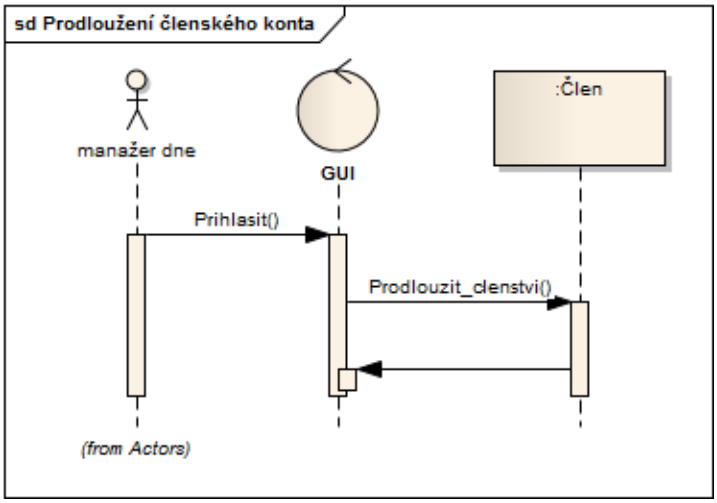
4.5.2. **Realizace případů použití**

Tyto diagramy slouží především pro jasné pochopení, jakým způsobem spolu komunikují vytvořené analytické třídy s případy užití. Na prvním diagramu je znázorněn postup při hlášení konce členství. Vše začíná zjištěním, že dochází ke konci období u některého z členů. Je tedy vytvořeno hlášení, které se zobrazí pop ohlášení do systému manažerovy dne, který si toto hlášení následně může otevřít a zjistit, který členský účet končí.



Obrázek 4.15 Sekvenční diagram pro hlášení konce členství

Další sekvenční diagram zachycuje, jakým způsobem dochází k prodloužení členského účtu. Po přihlášení manažer dne je schopen prodloužit členské konto pomocí grafického rozhraní.



Obrázek 4.16 Sekvenční diagram pro Prodloužení členského konta

4.6. Návrh řešení dalšího postupu

V tuto chvíli je vytvořena analýza nového informačního systému a celý projekt se nachází ve fázi rozpracování.

V době dokončování diplomové práce se začíná plně pracovat na přeměně analytických tříd na návrhové a vytvářejí se další diagramy, které jsou součástí návrhu.

V projektu bylo rozhodnuto, že budou zachovány oba diagramy, tedy jak diagram analytických tříd, tak návrhový diagram tříd. Bylo tak rozhodnuto s ohledem na velikost systému. Jakmile se do diagramu tříd začnou zapisovat podrobné informace o všech metodách a přibudou navíc další třídy, které budou sloužit pro komunikaci, začne být celý model nepřehledný.

Další věcí, o které se uvažuje, při přechodu na návrhové třídy je rozdělení diagramu analytických tříd s názvem účet na dva diagramy. To by bylo dosaženo oddělením třídy položka účtu a jejími návaznostmi do nového diagramu.

Při vytváření analýzy bylo zjištěno, že třída účet je propojena s velkým počtem dalších tříd, jako jsou: rezervace, členské účty, záznamy o platbách, položkách účtu a dalších. Bude tedy nutné postupovat při další implementaci nanejvýš opatrně a zohlednit toto riziko i při tvorbě návrhového vzoru tříd.

Druhou složitější třídou je třída položka účtu, která je přímo spojena s výše zmiňovaným účtem a také je spojena s velkým počtem tříd.

Tyto třídy se zatím nemusí zdát jako příliš problémové, problém však může nastat, při vytváření pohledu z databáze přes více tříd. Pokud bude databáze obsahovat několik desítek tisíc záznamů a nebude správně navržený databázový model. Nemohla by být garantována reakční doba při výpisu nebo i práci s těmito třídami.

5. Zhodnocení a závěr

5.1. Zhodnocení

Vytvořením analýzy a následnému doporučení dalšího postupu při návrhu systému se celý projekt dostává do fáze tvorby samotného počítačového programu.

Díky použití metody Unified process je zajištěno, že pokud bude potřeba v průběhu následující fáze vydefinovat další požadavky, což se u projektu této velikosti dá předpokládat, nebude jejich implementace příliš velkým problémem. Navíc se dá dobře sledovat vývoj softwaru, díky rozdělení projektu na menší části.

Při sběru požadavků se ukázalo, že vytváření případů užití je velmi dobrým nástrojem pro zachycení funkčnosti. Zaměstnanci po malé instruktáži bez problémů pochopili hlavní myšlenku případů užití a dokázali hledat v diagramu, zda obsahuje požadovanou funkčnost.

Naopak analytické třídy jsou určeny techničtějším uživatelům, proto byly konzultovány především s vedoucím pracovníkem IT a programátorem projektu.

Celá fáze sběru požadavků a analýza proběhla bez větších problémů, díky čemuž tato fáze nijak neohrozí prodlení projektu.

5.2. Závěr

Cílem této diplomové práce bylo objasnit problematiku tvorby informačních systémů se zaměřením na postup vývoje, sběr požadavků na systém a analýzu. Následně bylo cílem vytvořit soupis požadavků a analýzu nově vznikajícího systému.

V praktické části byly sepsány všechny požadavky, které byly kladeny na nový informační systém. Požadavky byly sbírány formou konzultace s hlavními uživateli stávajícího informačního systému a zkoumáním stávajícího řešení. Požadavky byly rozděleny na funkční a nefunkční a dále pro větší přehlednost tříděny do dalších.

Na základě těchto požadavků byly vydefinovány aktéři systému a stanoveny případy užití, které slouží pro zpřehlednění funkcionality systému a pro zpětnou kontrolu nasbíraných požadavků od zákazníka. Pomocí těchto případu užití je možné i pro nezainteresovanou osobu pochopit funkčnost systému. Aktérů v systému bylo vydefinováno celkem osm. Tito

aktéři byli rozděleni do tří skupin. První skupinou jsou uživatelé systému, kteří se starají o provoz v podniku, druhá skupina obsahuje uživatele zařizující obchod a vedení podniku a třetí skupina jsou uživatelé, starající se o aplikaci. Diagramů případů užití bylo vytvořeno celkem šest. Tyto diagramy obsahují vždy případy užití z nějaké skupiny požadavků. Dle požadavků byly vydefinovány tyto skupiny: rezervace, sklad, správa systému, správa účtu, správa členů a řízení a přehledy.

Po odsouhlasení diagramů případů užití byla započata analýza systému, obsahující především vytvoření analytických tříd systému a vytvoření modelů interakce mezi analytickými třídami a případy užití. Diagramy analytických tříd byly také rozděleny do skupin, které obsahují vždy určitý rámec tříd systému. Vznikly diagramy pro členy, počítače, zaměstnance a nejkomplikovanější diagram pro účet systému.

Z analýzy vyplývá, že nejproblémovější bude implementace třídy účet a jejího okolí. Jde o třídu, která se dá označit jako stěžejní v celé aplikaci a je napojena na velký počet dalších tříd. Dá se tedy očekávat, že při vytváření podrobného návrhu systému se bude muset věnovat velká pozornost implementaci této třídy.

Díky využití metodiky UP, která funguje na principu iterací, je zajištěno, že pokud by vznikly nové požadavky na systém v průběhu jeho vytváření, bude stále dost času na jejich zapracování.

V době psaní této diplomové práce se začíná zpracovávat dokument návrhu informačního systému a vytvářejí se první spustitelné kódy.

6. Seznam použitých zdrojů

6.1. Bibliografické zdroje

BASL, Josef; BLAŽÍČEK, Roman. *Podnikové informační systémy : Podnik v informační společnosti*. 2. vydání. Praha : Grada, 2008. 288 s. ISBN 978-80-247-2279-5.

BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů*: *kategorizace, agilní metodiky, vzory pro návrh metodiky*. 1. vyd. Praha: Grada, 2005, 163 s. ISBN 80-247-1075-7.

ELLIOTT, Geoffrey. *Global business information technology*: *an integrated systems approach*. New York: Pearson Addison Wesley, 2004, 503 s. ISBN 03-212-7012-6.

FOWLER, Martin. *Destilované UML*. 3. vydání. Praha : Grada, 2009. 176 s. ISBN 978-80-247-2062-3.

GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. *Podniková informatika*. 2., přeprac. a aktualiz. vyd. Praha: Grada, 2009, 496 s. Expert (Grada). ISBN 978-80-247-2615-1.

KANISOVÁ, Hana a Miroslav MÜLLER. *UML srozumitelně*. 2. aktualiz. vyd. Brno: Computer Press, 2006, 176 s. ISBN 80-251-1083-4.

LIU, Liping a Boris ROUSSEV. *Management of the object-oriented development process*: *June 19 - 23, 2000, Schaumburg, Illinois*. Hershey: Idea Group Pub., c2006, 372 s. ISBN 1-59140-606-4.

NEUSTADT, Ila; ARLOW, Jim. *UML 2 : a unifikovaný proces vývoje aplikací*. Brno : COMPUTER PRESS, 2007. 568 s. ISBN 978-80-251-1503-9.

RATZAN, Lee. *Understanding information systems*: *what they do and why we need them*. Chicago: American Library Association, 2004, 253 s. ISBN 08-389-0868-3.

SODOMKA, Petr; KLČOVÁ, Hana. Informační systémy : v podnikové praxi. 2. vydání. Brno : Computer Press, a.s., 2010. 504 s. ISBN 978-80-251-2878-7.

6.2. **Elektronické zdroje**

ABZ.CZ *ABZ.cz*: *slovník cizích slov* [online]. 2006 [cit. 2012-03-19]. Dostupné z: http://slovnik-cizich-slov.abz.cz

ASPECT WORKS *Aspect Works* [online]. 2012 [cit. 2012-03-26]. Dostupné z: http://www.aspectworks.com

BUCHALCEVOVÁ, Alena. Jakou metodiku použít pro konkrétní projekt?: Hodnocení a výběr vhodné metodiky pro budování IS. *SOFTEC* [online]. 2010 [cit. 2012-03-17]. Dostupné z: http://www.softec.sk/files/Softecon/Softecon2010/Buchalcevova.pdf

BLUE ORANGE A.S. *Blue Orange* [online]. 2003, 2012 [cit. 2012-03-19]. Dostupné z: http://www.blueorange.cz/

Co je to framework?. LÁNG, Peter. *Webárna* [online]. 2010 [cit. 2012-03-25]. Dostupné z: http://langi.cz/webarna/co-je-to-framework

Enterprise Architect: *Informační web o nástroji Enterprise Architec* [online]. 2008 [cit. 2012-03-19]. Dostupné z: http://www.enterprisearchitect.cz/

Extrémní programování v praxi. *Interval.cz* [online]. 2002 [cit. 2012-03-26]. Dostupné z: http://interval.cz/clanky/extremni-programovani-v-praxi

Fakulta informačních technologií. *ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE* [online]. 2012 [cit. 2012-03-25]. Dostupné z: http://www.fit.cvut.cz/student/bakalar/softwarove-inzenyrstvi

Introduction to OpenUP. BALDUINO, Ricardo. ECLIPSE. *Eclipse* [online]. 2007 [cit. 2012-03-28]. Dostupné z: www.eclipse.org/epf/general/OpenUP.pdf

KOLÁŘ, Petr. *Operační systémy* [online]. 2005 [cit. 2012-03-20]. Dostupné z: http://www.nti.tul.cz/~kolar/os/os.pdf

KŘENA, Bohuslav a Radek KOČÍ. Scribd. *Scribd* [online]. 2006. vyd. 2006 [cit. 2012-03-17]. Dostupné z: http://www.scribd.com/doc/48636337/23/Iterativni-model

MERUNKA, Vojtěch. METODY TVORBY INFORMAČNÍCH SYSTÉMŮ. *SMEP*: *systém mutimediální elektronické publikace* [online]. 2008 [cit. 2012-03-17]. Dostupné z: http://etext.czu.cz/img/skripta/64/pef_226-1.pdf

Objektový přístup. *INFORMAČNÍ TECHNOLOGIE* [online]. 2008 [cit. 2012-03-17]. Dostupné z: http://informacni-technologie.studentske.cz/2008/11/objektov-pstup.html

UML tools for software development and modelling: Enterprise Architect UML modeling tool. *Sparx systems* [online]. 2000, 2012 [cit. 2012-03-19]. Dostupné z: http://www.sparxsystems.com

Životní cyklus informačního systému. *Fakulta informatiky Masarykovy univerzity* [online]. 2005 [cit. 2012-03-17]. Dostupné z: http://www.fi.muni.cz/~smid/mis-zivcyk.htm

7. Přílohy

7.1. Slovník pojmů

Pojem	Definice	význam
Návštěvník	Návštěvník z ulice, který nemá založený účet v centru.	Synonyma: žádná Homonyma: žádná
Člen	Registrovaný člen centra, který platí členské příspěvky.	Synonyma: žádná Homonyma: žádná
Zákazník	Souhrnné označení pro návštěvníka nebo člena.	Synonyma: klient Homonyma: žádná
Účet	Virtuální místo, na které se zapisují veškeré produkty a služby, které si zákazník objednal.	Synonyma: žádná Homonyma: žádná
Sortiment	Veškeré služby, produkty a zboží podniku.	Synonyma: portfólio Homonyma: žádná
Místa	Může být v restauraci stůl na squash kurt atd..	Synonyma: žádná Homonyma: žádná
Umístění účtu	Úsek, v kterém se nachází aktuálně účet klienta.	Synonyma: žádná Homonyma: žádná
Systém	Označení systému jako jednotky, která provádí určitou činnost,	Synonyma: žádná Homonyma: žádná
Zaměstnanec	Jednotné označení všech rolí zaměstnanců v systému.	Synonyma: žádná Homonyma: žádná
Manažer dne	Zodpovídá za provoz v daný den	Synonyma: žádná Homonyma: žádná
Vedoucí úseku	Řeší vzniklé problémy se zákazníky.	Synonyma: žádná Homonyma: žádná
Číšník	Obsluhuje zákazníky	Synonyma: žádná Homonyma: žádná
Recepční	Zaměstnanec pracující na recepci.	Synonyma: žádná Homonyma: žádná
Hotelová recepční	Zaměstnanec procující na recepci v hotelu.	Synonyma: žádná Homonyma: žádná
Beauty pokladní	Zaměstnanec oddělení v beauty.	Synonyma: žádná Homonyma: žádná
Pokladní HC	Zaměstnanec oddělení v health clubu.	Synonyma: žádná Homonyma: žádná
Trenér	Radí, jak postupovat při sportech klientům	Synonyma: žádná Homonyma: žádná
Barman	Míchá nápoje na baru, nezasahuje do systému.	Synonyma: žádná Homonyma: žádná

Pojem	Definice	význam
Obchodní zástupce	Stará se o klienty a nabízí nové služby.	Synonyma: žádná Homonyma: žádná
Kuchař	Vaří jídlo, nezasahuje do systému, objednávky dostává formou lístků, které vyjíždějí z tiskárny v kuchyni.	Synonyma: žádná Homonyma: žádná
Manažer	Vrchní vedení společnosti v taktickém a strategickém úseku.	Synonyma: žádná Homonyma: žádná
Konto:	Členské konto vedené společností.	Synonyma: Členský účet Homonyma: žádná
Karta	Karta používáná pro identifikaci člena.	Synonyma: Členská karta Homonyma: Kreditní karta
Společnost	Označení podniku jako celku.	Synonyma: Blue Orange a.s., zadavatel, komplex Homonyma: žádná
Pokladní terminál	Elektronická pokladna umístěná na každém oddělení.	Synonyma: Pokladna Homonyma: žádná
Administrátor	Role obsluhující systém.	Synonyma: Admin Homonyma: žádná
Stůl	Místo v restauraci, na kterém se nacházejí jednotlivé účty.	Synonyma: žádná Homonyma: žádná
Číselník	Seznam s okruhem různých voleb, který slouží pro podporu při vyplňování různých kolonek	Synonyma: žádná Homonyma: žádná
Úsek	Část společnosti, která zahrnuje určitý okruh služeb.	Synonyma: Sekce, oddělení Homonyma: žádná
Pracovník	Jednotné označení pro zaměstnance, kteří se přímo starají o zákazníky.	Synonyma: žádná Homonyma: žádná
Služba	Jde o nehmotný produkt společnosti jako je kadeřnické práce apod.	Synonyma: žádná Homonyma: žádná
Produkt	Označení pro cokoli, co společnost nabízí svým klientům.	Synonyma: žádná Homonyma: žádná
Cenové indexy	Nastavení ceny produktů na jednotlivých úsecích.	Synonyma: žádná Homonyma: žádná
Suroviny	Jde o nakoupené zboží primárně určené k tvorbě produktů společnosti.	Synonyma: žádná Homonyma: žádná
Zboží	Jde o věc, které je koupena za účelem dalšího prodeje, bez jakékoliv modifikace.	Synonyma: žádná Homonyma: žádná
Storno	Zrušení nějaké objednávky.	Synonyma: zrušit Homonyma: žádná
Směna	Určité skupina zaměstnanců, které tvoří personál na nějaký den.	Synonyma: žádná Homonyma: žádná

Tabulka 7.1 Slovník pojmů

7.2. Funkční požadavky

ID	uživatel	znění	priorita	typ
FUN_1	systém	Systém bude sledovat a hlásit konec členství klientů.	Možný	členové
FUN_2	vedoucí úseku	Systém bude umět přiřadit ke členům garanta.	Možný	členové
FUN_3	manažer dne	Systém bude moct zakládat, upravovat a mazat členské účty.	Nezbytný	členové
FUN_4	manažer dne	Systém bude umět prodlužovat členská konta.	Nezbytný	členové
FUN_5	manažer dne	Systém bude zaznamenávat platby za členské příspěvky.	Nezbytný	členové
FUN_6	pracovník	Systém bude umožňovat pohlížet data členů.	Nezbytný	členové
FUN_7	pracovník	Systém bude umožňovat zapisovat poznámky ke členům.	Nezbytný	členové
FUN_8	manažer dne	Systém bude umět vytvořit záznam o členské kartě.	Nezbytný	členové
FUN_9	manažer dne	Systém bude umožňovat více členských karet.	Nezbytný	členové
FUN_10	pracovník	Systém bude sledovat obsazenost jednotlivých pokojů.	Nezbytný	rezervace
FUN_11	pracovník	Systém bude nabízet přiřazení rezervace k členům.	Možný	rezervace
FUN_11	pracovník	Systém bude vyhledávat v rezervacích.	Nezbytný	rezervace
FUN_12	pracovník	Systém bude sledovat plnost jednotlivých stolů v restauraci.	Nezbytný	rezervace
FUN_13	pracovník	Systém bude sledovat obsazenost jednotlivých sálů.	Nezbytný	rezervace
FUN_14	pracovník	Systém bude vytvářet, upravovat nebo mazat rezervace na jednotlivých odděleních.	Nezbytný	rezervace
FUN_15	pracovník	Systém bude sledovat plnost zásob ve skladu.	Nezbytný	sklad
FUN_16	pracovník	Systém bude sledovat intenzitu odběru zásob ve skladu.	Eventuální	sklad
FUN_17	systém	Systém bude hlásit položky, u kterých dochází množství.	Nezbytný	sklad
FUN_18	vedoucí úseku	Systém bude exportovat a tisknout stav skladu.	Nezbytný	sklad
FUN_19	vedoucí úseku	Systém bude umět vytvořit jídelníček pro restauraci	Nezbytný	sklad

ID	uživatel	znění	priorita	typ
FUN_20	vedoucí úseku	Systém bude pracovat s jednotlivými složkami sortimentu.	Nezbytný	sklad
FUN_21	vedoucí úseku	Systém bude umět upravovat množství jednotlivých položek sortimentu.	Nezbytný	sklad
FUN_22	vedoucí úseku	Systém bude vytvářet nové surovin, zboží, produkty nebo služby.	Nezbytný	sklad
FUN_23	manažer	Systém bude nabízet různé sestavy vygenerované z programu.	Nezbytný	sledování
FUN_24	manažer dne	Systém bude zobrazovat a tisknout sestavy prodejů (buďto vše nebo dle druhu zboží nebo oddělení)	Nezbytný	sledování
FUN_25	manažer dne	Systém bude zobrazovat a tisknout přehledy účtů (buďto vše nebo dle slev, typu úhrady nebo zrušených účtů)	Nezbytný	sledování
FUN_26	manažer	Manažer dne si zobrazuje a tiskne sestavy prodejů (buďto vše nebo dle druhu zboží nebo oddělení)	Nezbytný	sledování
FUN_27	manažer	Systém bude zobrazovat a tisknout sestavy tržeb a přehledy účtů (buďto vše nebo dle slev, typu úhrady nebo zrušených účtů)	Nezbytný	sledování
FUN_28	obchodní zástupce	Systém bude sledovat stavy jednotlivých účtů.	Nezbytný	sledování
FUN_29	obchodní zástupce	Systém bude nabízet různé sestavy pro různé role v systému.	Nezbytný	sledování
FUN_30	obchodní zástupce	Systém bude zasílat nepřečtené správy na výše uvedeného pracovníka.	Nezbytný	sledování
FUN_31	vedoucí pracovník	Systém bude třídit vygenerované sestavy.	Nezbytný	sledování
FUN_32	pracovník	Systém bude umět měnit heslo u jednotlivých pracovníků.	Nezbytný	správa
FUN_33	administrátor	Systém bude vyhledávat, vypisovat, přidávat, upravovat nebo mazat tiskárny ze systému.	Nezbytný	správa
FUN_34	administrátor	Systém bude vyhledávat, vypisovat, přidávat, upravovat nebo mazat skupiny uživatelů ze systému.	Nezbytný	správa
FUN_35	administrátor	Systém bude přiřazovat jednotlivé zaměstnance do skupin s oprávněním.	Nezbytný	správa
FUN_36	administrátor	Systém bude vyhledávat, vypisovat, přidávat, upravovat nebo mazat zaměstnance ze systému.	Nezbytný	správa
FUN_37	administrátor	Systém bude umět spravovat seznam s číselníky.	Nezbytný	správa

ID	uživatel	znění	priorita	typ
FUN_38	administrátor	Systém bude vyhledávat, vypisovat, přidávat, upravovat nebo mazat magnetické čtečky ze systému.	Nezbytný	správa
FUN_39	administrátor	Systém bude upravovat cenové indexy.	Nezbytný	správa
FUN_40	administrátor	Systém bude zálohovat celé nastavení systému.	Nezbytný	správa
FUN_41	administrátor	Systém bude moci načíst zálohu celého systému.	Nezbytný	správa
FUN_42	administrátor	Systém bude vyhledávat, vypisovat, přidávat, upravovat nebo mazat počítače ze systému.	Nezbytný	správa
FUN_43	administrátor	Systém bude přiřazovat k počítačům tiskárny a čtečky.	Nezbytný	správa
FUN_44	administrátor	Systém bude zařazovat nové počítače do systému.	Nezbytný	správa
FUN_45	administrátor	Systém bude umožňovat roční zálohu aplikace.	Nezbytný	správa
FUN_46	administrátor	Systém bude umožňovat nastavení číselníků systému.	Nezbytný	správa
FUN_47	pracovník	Systém bude zakládat, upravovat a uzavírat účet zákazníka včetně jeho zaplacení.	Nezbytný	účet
FUN_48	pracovník	Systém bude umožňovat převést účet na jiné oddělení.	Nezbytný	účet
FUN_49	pracovník	Systém bude umožňovat platit v restauraci stravenkou.	Nezbytný	účet
FUN_50	pracovník	Systém bude umožňovat platby následujícími způsoby: (platba kupónem, hotovostí, platby kartou, platba na fakturu, kreditem, slevová poukázka).	Nezbytný	účet
FUN_51	pracovník	Systém bude umět uzavřít pokladu.	Nezbytný	účet
FUN_52	pracovník	Systém bude umožňovat rozdělení účtu na více plateb.	Nezbytný	účet
FUN_53	pracovník	Systém bude umět vyhledat účet včetně oddělení.	Nezbytný	účet
FUN_54	pracovník	Systém bude umět převádět účty na různě stoly.	Nezbytný	účet
FUN_55	pracovník	Systém umožňuje vkládat libovolné položky ze sortimentu.	Nezbytný	účet
FUN_56	pracovník	Systém bude umět tisknout zadané jídlo do systému.	Nezbytný	účet
FUN_57	pracovník	Systém bude umožňovat objednání jídla na účet vedený na jiném oddělení.	Nezbytný	účet
FUN_58	pracovník	Systém bude umožňovat zadávat jídlo	Nezbytný	účet

ID	uživatel	znění	priorita	typ
		sebou.		
FUN_59	pracovník	Systém bude přiřazovat účty v restauraci ke stolům.	Nezbytný	účet
FUN_60	pracovník	Systém bude podporovat více kuchyní v jedné restauraci (gril/vnitřní kuchyně)	Nezbytný	účet
FUN_61	pracovník	Systém bude podporovat na každém oddělení jiný sortiment.	Nezbytný	účet
FUN_62	pracovník	Systém bude umožňovat přesuny účtů mezi jednotlivými úseky.	Nezbytný	účet
FUN_63	pracovník	Systém bude umět z hotelového úseku objednat zboží z kteréhokoliv jiného úseku.	Nezbytný	účet
FUN_64	pracovník	Systém bude sledovat obsazenost jednotlivých pokojů v hotelu,	Nezbytný	účet
FUN_65	pracovník	Systém bude umět kontrolovat hosty hotelu v evidenci cizinecké policie.	Chceme mít	účet
FUN_66	pracovník	Systém bude umožňovat platit účet postupně.	Nezbytný	účet
FUN_67	pracovník	Systém bude tisknout účty.	Nezbytný	účet
FUN_68	pracovník	Systém bude umět hledat položky ze skladu.	Nezbytný	účet
FUN_69	pracovník	Systém bude umět hledat v seznamu uživatelů.	Nezbytný	účet
FUN_70	pracovník	Systém bude umět hledat v seznamu členů	Nezbytný	účet
FUN_71	pracovník	Systém bude umět měnit typ platby na účtu.	Nezbytný	účet
FUN_72	pracovník	Systém bude podporovat rezervaci z jednoho oddělení na druhé.	Nezbytný	účet
FUN_74	pracovník	Systém bude zajišťovat objednávku trenérů.	Nezbytný	účet
FUN_75	vedoucí úseku	Systém bude provádět storna na účtech klientů	Nezbytný	účet
FUN_76	vedoucí úseku	Systém bude vytvářet slevy na účtech klientů.	Nezbytný	účet
FUN_77	vedoucí úseku	Systému bude umět mazat položky na účtech klientů.	Nezbytný	účet
FUN_78	vedoucí úseku	Systém bude podporovat dědičnost rolí.	Nezbytný	účet
FUN_79	vedoucí úseku	Systém bude vytvářet slevy na účtech klientů.	Nezbytný	účet
FUN_80	pracovník	Systém bude zobrazovat rozpis směn.	Nezbytný	zaměstnanci
FUN_81	pracovník	Systém bude podporovat objednávky ze stran zaměstnanců.	Nezbytný	zaměstnanci

ID	uživatel	znění	priorita	typ
FUN_82	vedoucí pracovník	Systém bude vytvářet směny.	Nezbytný	zaměstnanci
FUN_83	vedoucí pracovník	Systém bude vytvářet přehledy směn.	Nezbytný	zaměstnanci
FUN_84	vedoucí pracovník	Systém bude vyhledávat a zobrazovat všechny zaměstnance	Nezbytný	zaměstnanci
FUN_85	vedoucí úseku	Systém bude vytvářet a upravovat rozpis služeb.	Nezbytný	zaměstnanci

Tabulka 7.1 Funkční požadavky

7.3. Scénáře pro případy užití

Případ použití: Hlásit konec členského konta	Případ použití: Prodloužit členské konto
ID: UC_29	**ID:** UC_30
Stručný popis: Systém zjistí konto, kterému v blízké době vyprší platnost.	**Stručný popis:** Prodlouží platnost členského konta.
Hlavní aktéři: Systém	**Hlavní aktéři:** Manažer dne
Vedlejší aktéři: Manažer dne	**Vedlejší aktéři:** Žádný
Vstupní podmínky: 1. Systém je spuštěn.	**Vstupní podmínky:** 1. Manažer dne je přihlášen. 2. Má vyhledaného člena viz UC_36. 3. Je splněn UC_34.
Hlavní scénář: 1. Případ použití začíná, jakmile existuje účet s blízkou skončením doby platnosti. 2. Systém zašle zprávu manažerovi dne	**Hlavní scénář**: 1. Manažer dne klikne na tlačítko prodloužit účet. 2. Systém nabídne volbu doby prodloužení. 3. Manažer dne zvolí dobu. 4. Manažer dne zapíše peněžní částku, kterou zákazník platí. 5. Systém napíše, kolik se má vrátit. 6. Manažer dne potvrdí celou transakci. 4. Systém prodlouží planost členského účtu.
Výstupní podmínky: 1. Systém zaslal zprávu.	**Výstupní podmínky:** 1. Vytištění potvrzení o prodloužení.
Alternativní scénář: Žádné.	**Alternativní scénář:** Žádné.
Případ použití: Založit členské konto	**Případ použití: Upravit členské konto**
ID: UC_31	**ID:** UC_32

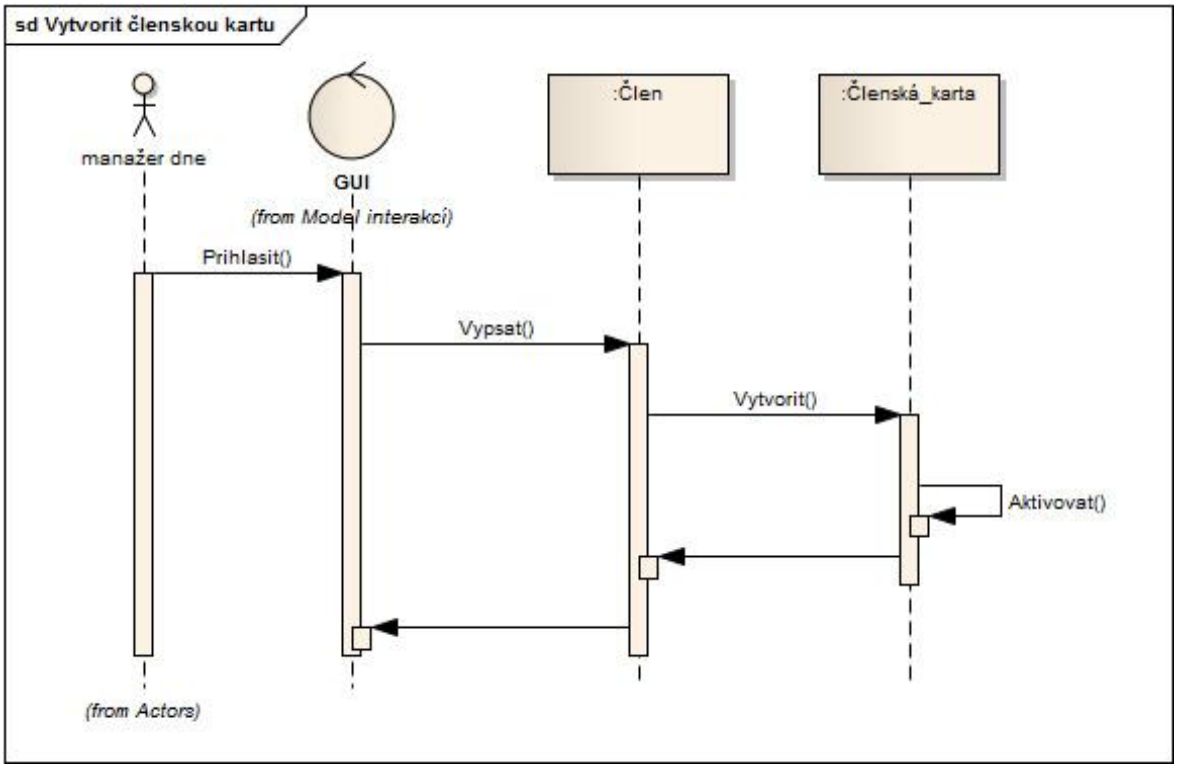
Stručný popis: Vytvoří nové členské konto.	**Stručný popis:** Upravuje jakékoliv zadané údaje u člena.
Hlavní aktéři: Manažer dne	**Hlavní aktéři:** Manažer dne
Vedlejší aktéři: Žádný	**Vedlejší aktéři:** Žádný
Vstupní podmínky: 1. Manažer dne je přihlášen.	**Vstupní podmínky:** 1. Manažer dne je přihlášen. 2. Má vyhledaného člena viz UC_36.
Hlavní scénář: 1. Manažer klikne na tlačítko založit nový uživatelský účet. 2. Systém vypíše formulář pro nového člena. 3. Manažer dne vypíše pole a odešle formulář. 4. Systém založí nový formulář.	**Hlavní scénář:** 1. Manažer klikne na tlačítko upravit uživatelský účet. 2. Systém vypíše formulář pro upravení člena 3. Manažer dne vypíše pole a odešle formulář. 4. Systém upraví data u člena
Výstupní podmínky: Žádná	**Výstupní podmínky:** 1. Uživatelská data jsou upravena
Alternativní scénář: Žádné.	**Alternativní scénář:** Žádné.
Případ použití: Zrušit členské konto	**Případ použití: Vytvořit členskou kartu**
ID: UC_33	**ID:** UC_38
Stručný popis: Smaže záznam z databáze o členovy.	**Stručný popis:** Vytvoří záznam o nové členské kartě.
Hlavní aktéři: Manažer dne	**Hlavní aktéři:** Manažer dne
Vedlejší aktéři: Žádný	**Vedlejší aktéři:** Žádný
Vstupní podmínky: 1. Manažer dne je přihlášen. 2. Má vyhledaného člena viz UC_36	**Vstupní podmínky:** 1. Manažer dne je přihlášen. 2. Má vyhledaného člena viz UC_36
Hlavní scénář: 1. Manažer klikne na tlačítko smazat uživatelský účet. 2. Systém vypíše nové okno s potvrzením volby. 3. Manažer dne vypíše pole a odešle formulář. 4. Systém upraví data u člena.	**Hlavní scénář:** 1. Manažer dne klikne na tlačítko vytvořit nového člena. 2. Systém vypíše formulář pro upravení člena 3. Manažer dne zapíše údaje a klikne na tlačítko uložit. 4. Systém nabídne doby členství. 5. Manažer dne zvolí dobu. 6. Manažer dne zapíše peněžní částku, kterou zákazník platí. 7. Systém napíše, kolik se má vrátit. 8. Manažer dne potvrdí celou transakci.

	9. Systém prodlouží planost členského účtu.
Výstupní podmínky: 1. Uživatel je smazán.	**Výstupní podmínky:** 1. Systém zapsal k účtu číslo přiřazené karty.
Alternativní scénář: Žádné.	**Alternativní scénář:** Žádné.
Případ použití: Vyhledat člena	**Případ použití: Prolížet data členů**
ID: UC_36	**ID:** UC_37
Stručný popis: Vyhledá člena z jejich databáze.	**Stručný popis:** Vypíše údaje členů, bez možnosti jejich úpravy.
Hlavní aktéři: Pracovník Trenér	**Hlavní aktéři:** Pracovník Trenér
Vedlejší aktéři: Žádný	**Vedlejší aktéři:** Žádný
Vstupní podmínky: 1a. Pracovník je přihlášen. 1b. Trenér je přihlášen.	**Vstupní podmínky:** 1a. Pracovník je přihlášen. 1b. Trenér je přihlášen. 2. Má vyhledaného člena vzi UC_36.
Hlavní scénář: 1. Pracovník nebo trenér vypíše do formuláře pro hledání jméno nebo příjmení. 2. Systém vypíše všechny shody seřazené dle příjmení.	**Hlavní scénář:** 1. Pracovník nebo trenér klikne na tlačítko zobrazit detail členského účtu. 2. Systém vypíše data požadovaného člena. 3. Pracovník nebo trenér zapíše, zavře pohled tlačítkem zavřít.
Výstupní podmínky: 1. Systém vypsal člena	**Výstupní podmínky:** Žádné.
Alternativní scénář: 1. Pracovník nebo trenér klikne na ikonu abecedního seznamu uživatelů. 2. Systém vypíše všechny členy seřízené dle příjmení.	**Alternativní scénář:** Žádné.
Případ použití: Zapsat poznámku u člena	
ID: UC_37	
Stručný popis: Zapíše poznámku u vybraného člena.	
Hlavní aktéři: Pracovník Trenér	
Vedlejší aktéři: Žádný	
Vstupní podmínky: 1a. Pracovník je přihlášen 1b. Trenér je přihlášen 2. Má vyhledaného člena viz UC_36	

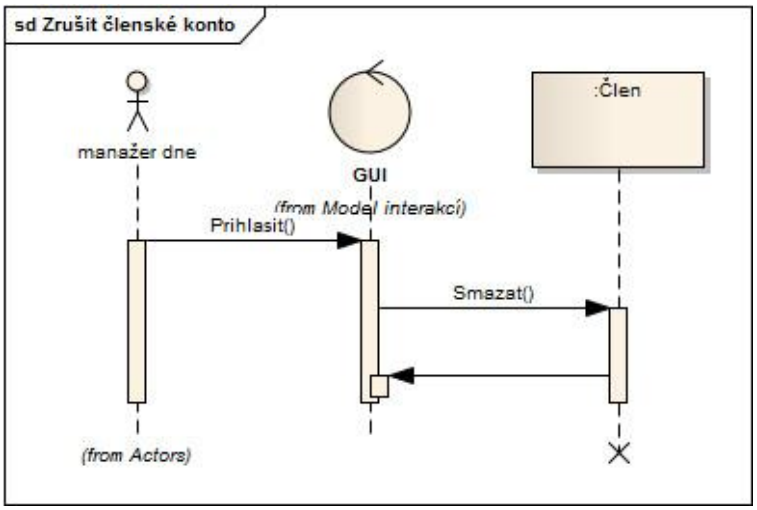
Hlavní scénář: 1. Pracovník nebo trenér klikne na tlačítko upravit členský účet. 2. Systém vypíše formulářovou kolonku s minulýma poznámka a s možností zapsaní nové. 3. Pracovník nebo trenér zapíše poznámku do systému a klikne na tlačítko uložit. 4. systém uloží poznámku u člena.
Výstupní podmínky: 1. Systém uložil novou poznámku
Alternativní scénář: Žádné.

Tabulka 7.2 Scénáře užití pro členy v systému

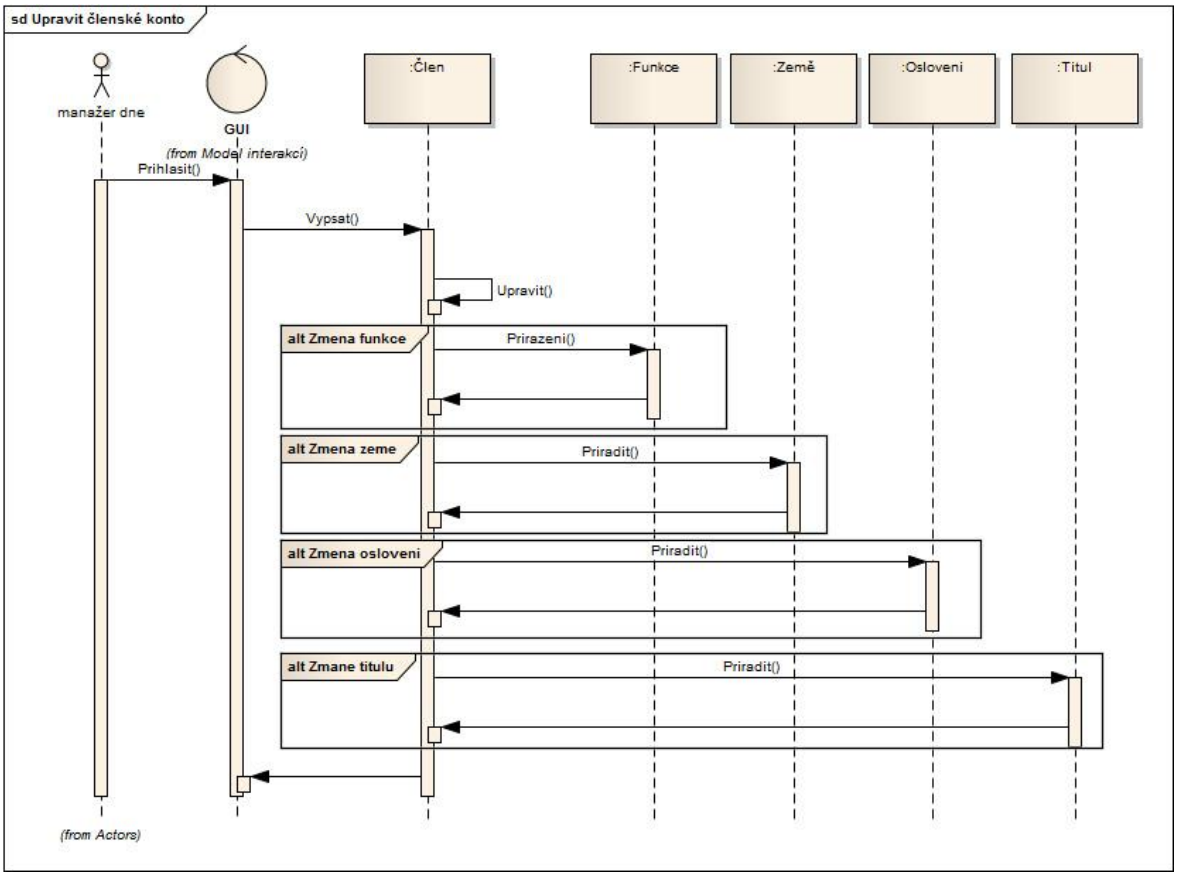
7.4. Sekvenční diagramy



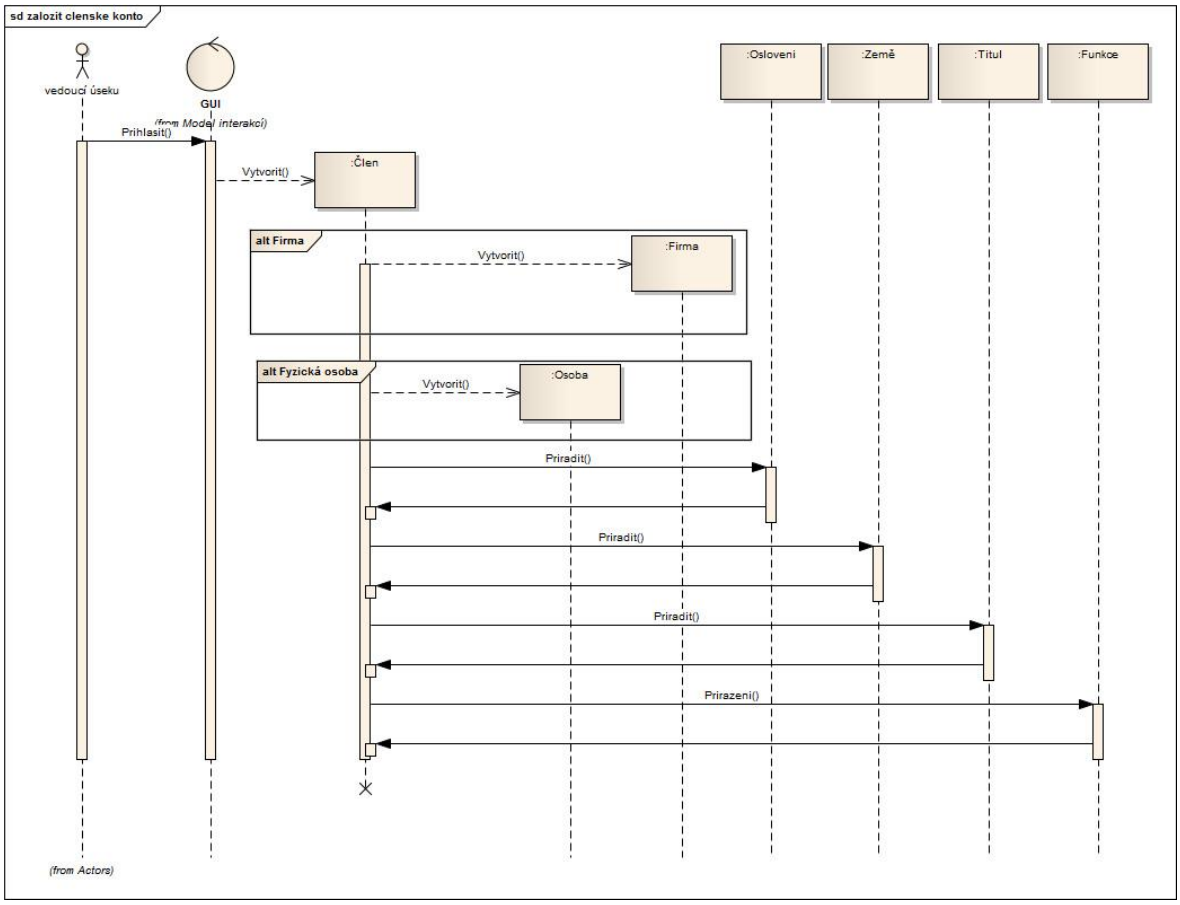
Obrázek 7.3 sekvenční diagram tvorba členské karty



Obrázek 7.4 sekvenční diagram zrušení členského konta



Obrázek 7.5 sekvenční diagram úprava členského konta



Obrázek 7.6 sekvenční diagram založení členského konta

7.5. Seznam obrázků

7.6. **Seznam tabulek**