Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

# Bakalářská práce

# Specifikace požadavků na software ve formátu případů užití

Plzeň 2014 Lukáš Týnovský

(Zádání)

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. 6. 2014 …….

Lukáš Týnovský

Abstract

The bachelor thesis deals with the specification of software requirements, specifically using the format of use cases and their template defined by Alistair Cockburn. The first part of the thesis is dedicated to the necessary theory of requirements specification, the Software requirements specification document and its sections, the use cases and their modeling in UML diagrams and in software. The second part of the thesis includes the process of creating the use cases of the system SPOT - in translation "*The dictionary system of technical terminology*" and the detail analysis of the use cases specification from the project CoCoME - *The Common Component Modeling Example* not only according to Alistair Cockburn's recommendations but also to author's experiences based on literature survey. The Annex contains the SPOT use cases and UML diagram and analyzed use cases.

Abstrakt

Bakalářská práce se zabývá specifikací softwarových požadavků, konkrétně pomocí formátu případů užití a využívá pro ně šablony Alistaira Cockburna. V první části se práce věnuje potřebné teorii specifikace, dokumentu specifikace požadavků a jeho částí, případů užití a jejich modelování pomocí UML diagramů a pomocí softwaru. Druhá část práce obsahuje postup tvorby případů užití systému SPOT - *Slovník překladů odborné terminologie* a podrobnou analýzu případů projektu CoCoME - *The Common Component Modeling Example*, jak s přihlédnutím ke zmíněným doporučení Alistaira Cockburna, tak s využitím zkušeností získaných studiem další literatury. Přílohou pak jsou vytvořené případy užití a UML diagram systému SPOT a analyzované případy užití.

# Obsah

# 1. Úvod

Specifikace softwarových požadavků se používá při tvorbě a údržbě softwarových produktů. Jedná se o popis požadovaného chování a vlastností výsledného programu. Dostatečná dokumentace požadavků umožňuje snazší rozšiřování a hledání směru dalšího vývoje aplikace.

Specifikace požadavků je důležitým pojítkem mezi zadavatelem softwaru, často IT laikem, a vývojáři softwaru, což bývají naopak úzce počítačově specializovaní lidé. Specifikace musí jasně definovat výsledný produkt. Úspěch produktu tak často závisí více na správném vzájemném porozumění a komunikaci tvůrců se zákazníkem, než na samotné kvalitě programování produktu. Vývojový tým musí při nedostatečně podrobné specifikaci spoléhat na zdravý rozum a znalost potřeb zákazníka. Navíc oprava chyb nebo nedorozumění, vzniklých v průběhu definice požadavků, které jsou odhaleny až při dalších fázích vývoje (návrh, programování, testování, produkční nasazení), je finančně a časově výrazně náročnější. To může výsledný produkt významně prodražit. Z toho důvodu je velice důležité nepodceňovat specifikaci požadavků a jejich správnou analýzu.

Cílem práce bylo seznámit se s hlavními způsoby specifikace požadavků softwaru a s vybranými nástroji pro jejich modelování a následně vytvořit několik případů užití (use cases) pro již existující aplikace. Tyto případy užití měly být využity k jejich poloautomatické analýze.

Práce obsahuje úvod do softwarových požadavků, jejich typů a specifikací, popis dokumentu specifikace požadavků a seznámení se s případy užití a jejich obsahem. Dále v práci nalezneme popis nástrojů pro modelování požadavků a popis vlastní práce. V příloze nalezneme samotné vytvořené případy užití.

# 2. Specifikace požadavků softwaru

V úvodní kapitole bylo vysvětleno, proč je správná specifikace požadavků softwaru klíčová pro vývoj aplikací. V této kapitole je popsáno, co takový požadavek vůbec je, jaké jsou jeho typy a jak se požadavky definují v dokumentu specifikace požadavků a jak a z čeho je tvořen případ užití (use case).

## 2.1 Definice požadavku

Definic požadavku je celá řada. Požadavkem může být cokoliv, co ovlivňuje rozhodování při návrhu [Lawrence 1997]. Standardizovaný slovník terminologie softwarového inženýrství Institutu pro elektrotechnické a elektronické inženýrství [IEEE, 1990] definuje požadavek jako:

- Podmínka nebo schopnost, nutná pro vyřešení problému uživatele nebo dosažení cíle.
- Podmínka nebo schopnost, která musí být splněna systémem nebo jeho komponentou k vyhovění smlouvě, standardu, specifikace nebo dalším vztahujícím se dokumentům.
- Zdokumentovaná reprezentace předchozích bodů.

Karl E. Wiegers uvažuje o požadavku jako o vlastnosti, kterou produkt musí mít, aby přinášel zúčastněným subjektům (stakeholders) hodnotu [Wiegers 2003].

## 2.2 Sběr a analýza požadavků

Při jejich hledání požadavků by se mělo dbát na potřeby uživatelů a zákazníků, které by měli mít tvůrci specifikace stále na paměti. Proto je třeba nejdříve zjistit, kdo bude systém užívat a zjistit další *zúčastněné subjekty (stakeholders)* produktu. Poté je třeba určit a pochopit jejich potřeby. Ty by měly být následně rozděleny a detailněji popsány do jednotlivých kategorií typů požadavků. Doporučuje se je doplnit ohodnocenou prioritou požadavku.

Když jsou požadavky sepsány, je třeba je ověřit a zjistit, zda jsou přesné a zda se mezi nimi nevyskytují například požadavky zcela zbytečné, duplicitní či protichůdné. Případné rozpory je třeba samozřejmě vyřešit, nejlépe se zástupci *uživatelských rolí (user role)*, kterých se problém týká, nebo alespoň se zástupci zákazníka.

Výsledné požadavky by měly splňovat následující kritéria. Měly by být kompletně a přesně popsány, měly by být realizovatelné, nezbytné, prioritizované, jednoznačné a ověřitelné [Wiegers 2003].

Dnes se často používá postup agilních metodik, kdy tvorba a specifikace požadavků a následný vývoj probíhají cyklicky ve vývojových iteracích od nejdůležitějších požadavků na software v první iteraci po další rozšíření programu v iteracích dalších. Splnění požadavků a jejich úprava

pak probíhá po každé iteraci vývoje. Tím má být zajištěno stálé zaměření projektu na prioritní potřeby zákazníků, které se mohou i v průběhu vývoje měnit.

## 2.3 Typy požadavků

Požadavky je možno roztřídit do kategorií podle různých autorů různě, toto následující zvolil autor práce jako, podle jeho názoru, nejvhodnější.

### 2.3.1 Funkční požadavky

Funkční požadavky vyjadřují, co přesně a co všechno má daný softwarový produkt dělat. Jinými slovy jde o popis toho, jak má software reagovat na dané vstupy a co má být následnými výstupy programu.

Definují se větou „Systém musí…," [IEEE, 1998] s doplněním odpovídajících vstupů a výstupů.

Tyto požadavky je možné dělit na podfunkce (subfunction) a podprocesy (subprocesses), což ale nemusí odpovídat softwarovému návrhu vnitřní struktury programu [IEEE, 1998].

Funkční požadavky tvoří důležitou část Dokumentu specifikace požadavků (DSP).

### 2.3.2 Požadavky na výkon

Požadavky na výkon jsou důležité zejména tam, kde má výstup funkce smysl pouze na omezenou dobu, například realtime systémy (např. řízení elektrárny) nebo tam, kde čekání na výstup stojí významné finanční prostředky.

V dnešní době však klademe požadavky na výkon téměř vždy, nikdo nechce na nic dlouho čekat a ztrácet tak zbytečně čas.

Požadavky na výkon se nejčastěji vyjadřují v časových jednotkách s pravděpodobnostní hodnotou, to znamená například „90 % požadavků na uložení dat do systému bude dokončeno do 1 vteřiny."

Dále je možné požadavek na výkon vyjádřit pracovní zátěží (Workload), kterou by měl software zvládnout, například „Webový portál musí zvládnout obsloužit 1000 návštěvníků za hodinu." To je možné rozvinout o tabulku předpokládaných scénářů návštěvy, aby vývojový tým věděl, jaké konkrétní požadavky zahrnuje požadovaný počet uživatelů.

### 2.3.3 Požadavky na vnější rozhraní

Požadavky na vnější rozhraní je možné rozdělit na požadavky na uživatelské prostředí, na hardwarové rozhraní, na softwarové rozhraní a na komunikační rozhraní.

#### *Uživatelské prostředí*

Zahrnuje požadavky na rozložení a obsah uživatelského prostředí na obrazovce a případně i podrobně na jednotlivé obrazovky a průchod jimi.
Někdy se mezi tyto požadavky uvádí podporované periferie počítače, které by ale měly patřit do hardwaru.

#### *Hardwarové rozhraní*

Obsahuje detailní popis konfigurace hardwaru zařízení, na kterém má daný systém fungovat. Často je zde uváděn i operační systém daného stroje, ač jde o software.

#### *Softwarové rozhraní*

Jedná se o soupis softwaru, se kterým by systémem měl pracovat, například databázové servery, překladače programovacích jazyků a další podpůrné programy. Někdy zde bývají v souvislosti s překladači vymezeny programovací jazyky, ve kterých má být systém vytvořen.

#### *Komunikační rozhraní*

Do požadavků na komunikační rozhraní patří například parametry internetového připojení, ke kterému bude systém připojen, další podporované komunikační technologie (Bluetooth, Wifi,…) a využívané komunikační protokoly. Případně další požadavky, s jakými systémy má software komunikovat.

### 2.3.4 Omezení

Omezení vyjadřují seznam všech věcí, kterými je vývojář softwaru limitován při tvorbě systému. Může sem patřit i požadovaný programovací jazyk, licenční omezení, bezpečnostní, regulační a legislativní omezení a podobně.

### 2.3.5 Vlastnosti

Je množství vlastností, které mohou být požadovány po softwaru. Měly by být specifikovány tak, aby je bylo možné ověřit. Následuje seznam příkladů vlastností podle IEEE [IEEE, 1998].

#### *Spolehlivost a dostupnost*

Chování softwaru by mělo být spolehlivé a nemělo by docházet k neočekávanému chování aplikace. Všechny výjimky a funkce by měly být ošetřeny proti neočekávaným chybovým hlášením. Software by měl v průběhu času fungovat spolehlivě a měl by být dostupný ideálně v režimu 24/7 (24 hodin 7 dní v týdnu). Opět se často využívá procentuálního vyjádření dostupnosti a chybovosti, které je následně měřeno.

Může být též definován čas, kdy bude docházet ke správě a údržbě systému a kdy nebude systém k dispozici, případně postup, jak bude objednateli softwaru sdělena plánovaná odstávka nebo oprava systému.

Do této kategorie také může patřit řešení záloh systému.

#### *Zabezpečení*

Software by měl být zabezpečen proti úniku a neoprávněné manipulaci s daty, někdy dokonce proti fyzické krádeži hardwaru, na kterém jsou data systému uložena. Zároveň by mělo být zamezeno, aby měli uživatelé přístup i k jiným datům, než která potřebují ke své práci.

Měly by být definovány používané šifrovací a hashovací metody, přístupové role a jejich ověřování, tvorba záznamů (logů) změn dat a kontrola integrity systému.

#### *Udržovatelnost*

Systém by měl podporovat změny parametrů, které se v průběhu času mohou měnit (například sazba DPH). Software by neměl vyžadovat přílišnou manuální údržbu, vyžadující interakci s uživatelem. Vnitřní struktura systému by měla umožňovat snadné změny v kódu systému a celkově přehlednost zdrojového kódu a jeho struktury. Celková údržba, oprava a aktualizace softwaru by měly být finančně a časově co nejméně náročné. Při tvorbě softwaru by se tedy mělo pamatovat na možné změny, které mohou v průběhu provozování systému nastat. Ze stejného důvodu by měly být kladeny zvýšené požadavky na kvalitu a rozsah dokumentace softwaru.

#### *Přenositelnost*

Přenositelnost je požadavek na schopnost systému, být provozován na jiném hardwarovém zařízení. Přenositelnost je definována parametry dalších systémem podporovaných zařízení, na které by bylo případně možné systém přesunout. Parametry může být operační systém, podporované překladače programovacích jazyků a další softwarové a hardwarové specifikace podporovaného zařízení.

### 2.3.6 Další požadavky

Mezi další požadavky se obvykle řadí například požadavky na databázi systému a na další součásti systému, které může být výhodnější definovat odděleně od požadavků na celou aplikaci. Také sem patří požadavky na dokumentaci softwaru a současně samozřejmě vše, co nespadá do žádné jiné kategorie požadavku.

### 2.3.7 Metoda FURPS+

Metoda FURPS+ je požívána pro hodnocení kvality softwarových produktů, nicméně toto hodnocení probíhá na základě plnění jednotlivých typů požadavků na software. Požadavky této metody jsou rozděleny na Funkčnost (Functionality), Použitelnost (Usability), Spolehlivost (Reliability), Výkon (Performance) a Udržovatelnost (Supportability) a pod znakem + se skrývají další požadavky. Jedná se tedy o možný způsob rozdělení požadavků na jednotlivé typy.

## 2.4 Dokument specifikace požadavků

Dokument specifikace požadavků je ucelený popis požadovaného nebo již existujícího softwaru. Jeho struktura je dána IEEE standardem 830-1998, případně je od tohoto standardu odvozována. Pokud se některá z položek standardu projektu netýká, doporučuje se danou položku struktury v dokumentu nechat a tuto skutečnost označit [Wiegers 2003].

Dobře provedený dokument specifikace požadavků by měl být správný, jednoznačný, kompletní, konzistentní, ohodnocený důležitostí a/nebo stabilitou, ověřitelný, upravitelný a kontrolovatelný [IEEE, 1998].

### 2.4.1 Struktura podle IEEE std 830-1998

Níže naleznete strukturu Dokumentu specifikace požadavků s českým překladem jednotlivých kapitol dokumentu.

1. Introduction (Úvod)

    1.1 Purpose (Účel)

    1.2 Scope (Rozsah)

    1.3 Definitions, acronyms, and abbreviations (Definice, pojmy, zkratky)

    1.4 References (Odkazy)

    1.5 Overview (Přehled)

2. Overall description (Obecný popis)

    2.1 Product perspective (Produkt v souvislostech)

    2.2 Product functions (Funkce produktu)

    2.3 User characteristics (Charakteristiky uživatel)

    2.4 Constraints (Omezení)

    2.5 Assumptions and dependencies (Předpoklady a závislost)

3. Specific requirements (Specifické požadavky)

Appendixes (Přílohy)

Index (Rejstřík)

[IEEE, 1998]

### 2.4.2 Specifické požadavky

V kapitole specifické požadavky se detailně popisují jednotlivé typy požadavků. Tuto kapitolu je možné uspořádat různě, IEEE nabízí řadu možných rozdělení této části dokumentu podle režimů, uživatelských tříd, objektů, systémových vlastností, podnětů, hierarchie nebo kombinací předešlých. Některé šablony specifický požadavků najdete v příloze.

Pokud aplikace pracuje v různých režimech, při kterých je potřeba různých funkcí a rozhraní, doporučuje se specifické funkční požadavky a případně požadavky na externí rozhraní sdružit právě podle pracovních režimů softwaru.

3. Specifické požadavky
    3.1. Funkční požadavky
        3.1.1 Režim 1
            3.1.1.1 Požadavky na vnější prostředí
                3.1.1.1.1 Uživatelské rozhraní
                3.1.1.1.2 Hardwarové rozhraní
                3.1.1.1.3 Softwarové rozhraní
                3.1.1.1.4 Komunikační rozhraní
            3.1.1.2 Funkční požadavky
                3.1.1.2.1 Funkční požadavek 1
                .
                .
                .
                3.1.1.2.*n* Funkční požadavek *n*
            3.1.1.3 Výkon
        3.1.2 Režim 2
        .
        .
        .
        3.1.*m* Režim *m*
    3.2 Omezující podmínky
    3.3 Vlastnosti systému
    3.4 Další požadavky

*Rozdělení specifických požadavků podle režimů systému*[IEEE, 1998]

Požadavky je možno rozdělit podle uživatelských tříd, pokud je to vhodné pro přehlednost dokumentu. To se využívá zejména tam, kde se funkčnost aplikace výrazně liší pro jednotlivé uživatelské skupiny.

Stejně tak se dá sekce rozdělit podle logických objektů. Ty je možné určit libovolně, mohou sdružovat více fyzických objektů nebo i aktérů dohromady. Tyto objekty nemusí odpovídat programátorským objektům softwaru.

Je možné též kapitolu rozdělit podle požadovaných vlastností softwaru nebo podle podnětů, které vyvolají požadavky na funkce softwaru.

Další možností rozdělení je rozdělení funkcí hierarchicky. Je také možné využít kombinaci několika kritérií.

Specifické požadavky tvoří jádro celého dokumentu specifikace požadavků.

## 2.5 Případy užití (use cases)

Formou případů užití se specifikují funkční požadavky na software a částečně také požadavky na uživatelské rozhraní (například požadované položky formulářů a podobně).

Případ užití popisuje posloupnost interakcí mezi systémem a vnějším aktérem. Cílem případu užití by však nemělo být popsat chování systému, nýbrž jeho požadovanou funkčnost [Cockburn 2001]. Chování je popisem cesty, kterou daná funkce proběhne.

Aktér může být osoba nebo další softwarové nebo hardwarové zařízení [Cockburn 2001]. Někdy se místo pojmu *aktér (actor)* používá zobecněného slovního spojení *uživatelská role (user role)*.

Případy užití mají textovou podobu, ale souvislosti mezi nimi se dají zaznamenat graficky, například pomocí UML diagramů.

### 2.5.1 Příklad jednoduchého případu užití

Níže naleznete jednoduchý strukturovaný příklad případu užití, který popisuje chování systému při uživatelském požadavku na tisk dokumentu.

***Tisk z editoru dokumentů***

Rozsah: Prohlížeč dokumentů

Úroveň: Uživatelský cíl

Aktér: Uživatel

Kontext cíle: Uživatel si při práci s editorem dokumentů potřebuje dokument vytisknout.

Předpoklady: Uživatel má otevřený požadovaný dokument.

Spouštěč: Uživatel potřebuje vytisknout dokument z aplikace

Hlavní scénář:

1. Uživatel potřebuje vytisknout dokument a stiskne klávesy Ctrl + P.
2. Systém zobrazí okno s možností volby stránek k vytištění, tiskárny a s nastavením počtu kopií.
3. Uživatel zadá požadované parametry a stiskne tlačítko Tisk.
4. Systém odešle zvolenou část souboru a parametry tiskárně.

Rozšíření:

4a. Tiskárna není nalezena nebo není zapnutá

4a1. Uživatel je vyzván zapnout tiskárnu nebo vybrat jinou a opakovat krok 3.

### 2.5.2 Pravidla a praktické použití jednotlivých položek případu užití

Pravidla pro tvorbu případů užití jsou důležitá proto, aby se v případech užití vyznal nejen jejich autor, ale i vývojový tým, který má na jejich základě popisovaný systém vytvořit.

Autor by ale měl volit co nejvhodnější formu a strukturu případu užití pro danou problematiku a té by měl držet i u všech dalších případů užití daného systému tak, aby při procházení dokumentace bylo co nejsnazší a nejrychlejší jednotlivé případy užití číst a porovnávat mezi sebou.

Pokud případ užití odkazuje na další případ užití, je název druhého případu označen buď kurzívou, nebo podtržen [Cockburn 2001]. V ukázkových use casech je použito podtržení a pro větší přehlednost název případu obsahuje také jeho číselné označení.

Případ užití je možné více specifikovat a upřesnit přidáním další informačních položek, než je scénář. To nám umožní vytvořit a odlišit od sebe stejné scénáře, vytvořené však z jiného úhlu pohledu.

#### *Primární aktér (Primary actor)*

Jak název napovídá, jedná se o pojmenování primárního aktéra systému v daném scénáři. Primární aktér je ten, kdo má cíl, kterým se daný případ užití zabývá [Cockburn 2001].

Primárním aktérem obvykle bývá uživatel systému.

#### *Sekundární aktér (Secondary actor)*

Sekundárním aktérem mohou být další zainteresované osoby nebo oddělení, které zajišťují pro primárního aktéra nějakou činnost, stejně tak může být sekundárním aktérem hardware. Další

software může být sekundárním aktérem, jen pokud je mimo uvažovaný rozsah daného případu užití.

### *Rozsah (Scope)*

Definuje rozsah diskutovaného systému v rámci případu užití.

Obvykle bývá rozsahem míněn celý systém, pokud se nejedná o systém rozsáhlý, případně o případ užití, řešící uskupení systémů.

### *Kontext cíle (Goal in context)*

Sděluje souvislosti mezi systémem a daným případem užití, a také mezi jednotlivými případy užití.

### *Úroveň (Level)*

Úroveň pomáhá rozlišovat náhled a obecnost případu užití, zároveň pomáhá případ užití zařadit do souvislostí s ostatními případy. Získáme tak přehled o nadřazenosti a podřazenosti případu užití vzhledem k ostatním.

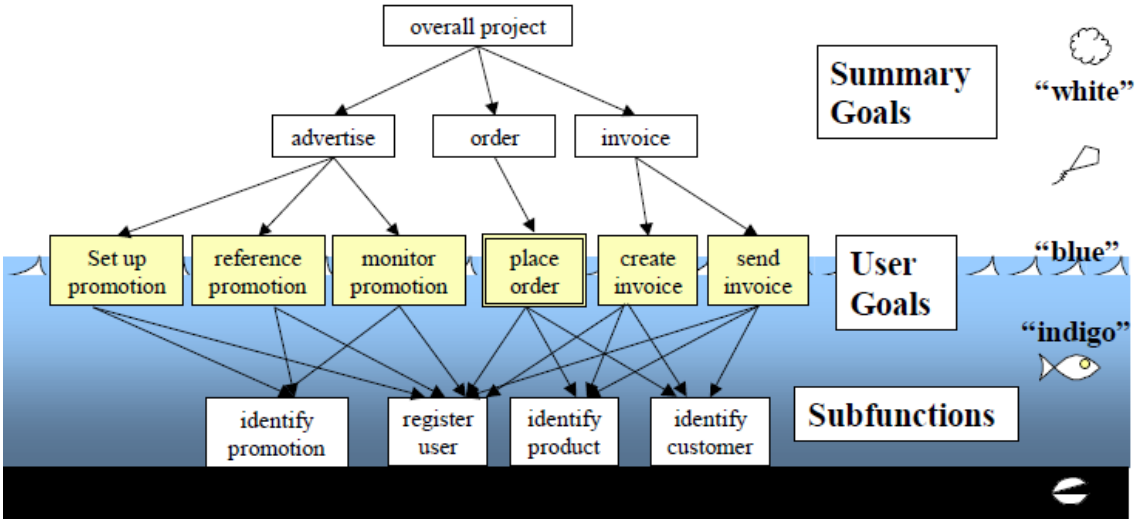
Neexistuje však žádné exaktně dané pravidlo, které by pomohlo vždy úroveň stoprocentně rozlišit, a tak záleží i na cítění autora, do jaké úrovně případ užití zařadí. Pak ale musí hrát roli kontext případu užití k ostatním případům.

Rozlišujeme 3 úrovně případu užití: Celkové cíle (Summary Goals), Uživatelské cíle (User Goals) a Podfunkce (Subfunctions), viz obrázek 1.

Celkové cíle jsou samozřejmě nadřazeny uživatelským a ty podfunkcím. K rozlišení úrovně lze použít řadu pomůcek, například otázku, zda si může po splnění cíle udělat primární aktér kávu [Cockburn 2001]. Pokud ano, pak se pravděpodobně jedná o uživatelskou úroveň. Případně může pomoci doba trvání scénáře do splnění cíle.

Protože se jedná o důležitou položku, bývá často použit symbol úrovně, umístěný na konec řádku s nadpisem případu užití.

Asi nejčastěji používanou úrovní je Uživatelský cíl.




*Obr. 1 Úrovně případů užití [Cockburn 2001]*

### *Spouštěč (Trigger)*

Spouštěč je událost, která bezprostředně předchází hlavnímu scénáři.

### *Předpoklad (Precondition) a Následná podmínka (Postcondition) a Garance (Guarantees)*

*Předpokladem* je míněno, co musí být splněno před spuštěním scénáře spouštěčem.

*Následná podmínka* nebo též *Garance* je podmínka, která musí být splněna po ukončení hlavního scénáře.

Obvykle se jedná o definice stavu nebo vlastností hlavního aktéra, ale může se také jednat o vlastnost nebo stav systému.

### *Hlavní scénář (Main success scenario)*

Hlavní scénář je samotná posloupnost interakcí, ovšem pouze pro případ, že vše probíhá v pořádku.

### *Rozšíření (Extensions)*

Jedná se o rozšíření hlavního scénáře, kde se uvádí jiný průběh, případně výsledek akcí. Čísla jednotlivých bodů rozšíření odkazují na body hlavního scénáře. Písmena pak rozlišují jednotlivé varianty rozšíření.

Obvykle se v rozšíření uvádí hlavně možné chyby, které mohou nastat při plnění hlavního scénáře, a reagování systému na ně.

### 2.5.3 Použití případů užití

Případy užití je možné použít k popisu požadavků v Dokumentu specifikace požadavků (DSP) nebo samostatně k popisu požadovaných funkcí systému. Obojího se využívá při vývoji softwaru.

V rámci DSP je možné případy užití použít jak v kapitole 2 – Obecný popis a použít obecnější use casy a UML diagramy, tak v kapitole 3 – Specifické požadavky, kde se použijí podrobnější případy užití na úrovni uživatelský cílů a podfunkcí.

Případy užití se využívají při vývoji softwaru při takzvaném Use case driven development, neboli vývoji řízeným případy užití, kdy se používají k detailnímu popisu funkcí systému a jsou postupně implementovány a testovány jednotlivě, či po scénářích (hlavní a pak alternativní a rozšířené).

Své použití najdou případy užití i při agilním vývoji aplikací, kde je možné je použít obdobným způsobem, ovšem zde nejsou psány tak dopodrobna a pro všechny funkce systému jako v předchozím případě, ale pouze pro ty funkce a do takové podrobnosti, jak je aktuálně třeba pro nejbližší vývoj, a detaily jsou přidávány až po implementaci předchozích základních verzí případů užití. Cílem zde není vytvořit dokonale srozumitelný případ užití, ale pouze zjednodušit komunikaci mezi vývojáři Může se tak stát, že některé takto vytvořené případy užití neobsahují všechny informace, potřebné k pochopení situace, ale jsou pouhou pomůckou k dalšímu vývoji.

Případy užití je také možné použít při plánování postupu vývoje, kdy se jednotlivé případy ohodnotí například podle důležitosti a náročnosti a podle toho se následně vytvoří posloupnost jednotlivých prací. To umožňuje dlouhodobé plánování a usnadňuje kontrolu, zda jde projekt stále stejným a správným směrem.

Někdy se také z případů užití vytváří takzvaná realizace případu užití (use-case realization), která například pomocí diagramu tříd nebo sekvenčního diagramu zachycuje části softwaru, které mají kroky dané případem užití vykonávat. To pomáhá ke snazšímu přechodu od případů užití k softwarovému produktu.

Pokud jsou případy užití správně strukturované, je možné je po úpravě použít k částečnému strojovému zpracování.

# 3. Nástroje pro modelování a specifikaci požadavků

V této kapitole naleznete popis nástrojů pro modelování softwarových požadavků, a to jak grafických, tak i softwarových.

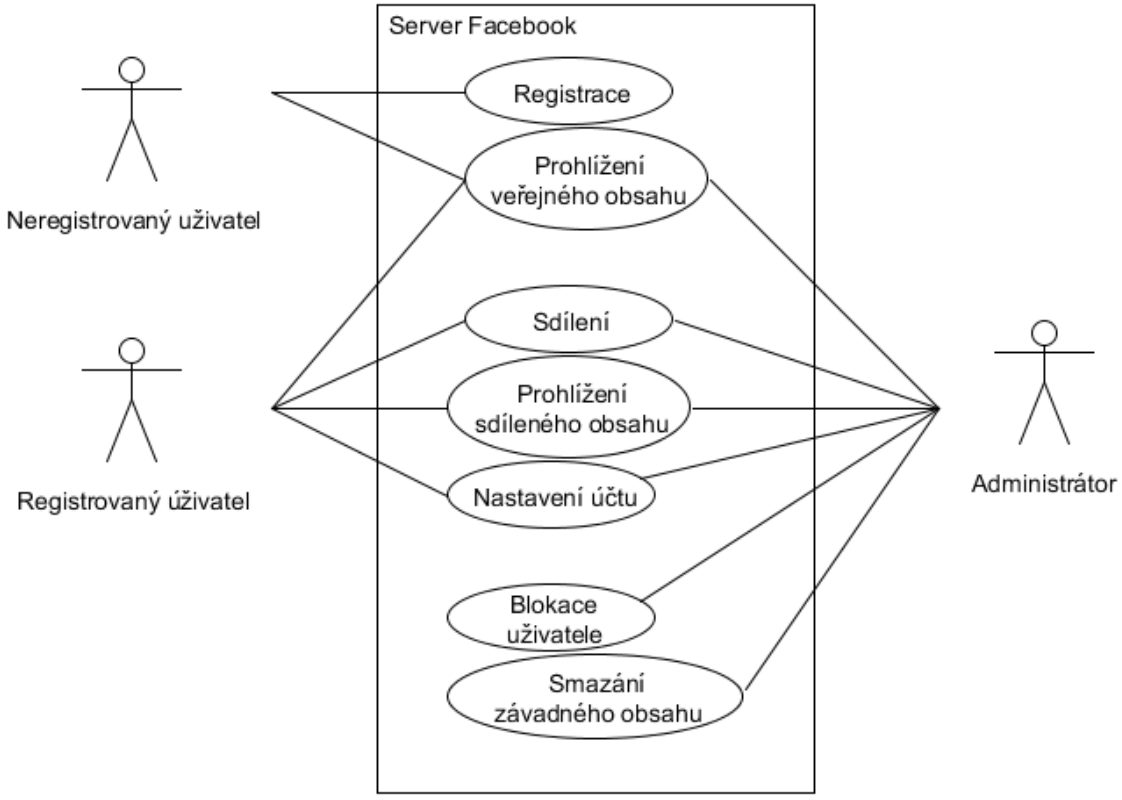
## 3.1 UML diagramy

UML, Unified Modeling Language (Jednotný modelovací jazyk) vznikl jako snaha sjednotit vizuální ztvárnění systému, obvykle software [Object Management Group 2013], jeho plánování, vizualizaci, dokumentaci a tvorbu.

UML 2.0 definuje 3 skupiny diagramů: Strukturální diagramy, Diagramy chování – ty obsahují diagramy užití (use case diagramy) a Diagramy interakce.

### 3.1.1 UML diagramy užití (use case diagramy)

Zobrazují aktéry nebo spíše jejich role v systému a jejich kontext s ním. Elipsa označuje případ užití, zatímco čárkovaná postavička označuje aktéra. Aktér může být ale podle UML standardu označen také čtvercem třídy s nápisem <<Actor>>, a pod ním název role, nebo ikonou počítače [Object Management Group 2013]. Příklad UML diagramu užití vidíte na obrázku 2.




*Obr. 2 UML diagram užití*

## 3.2 Software

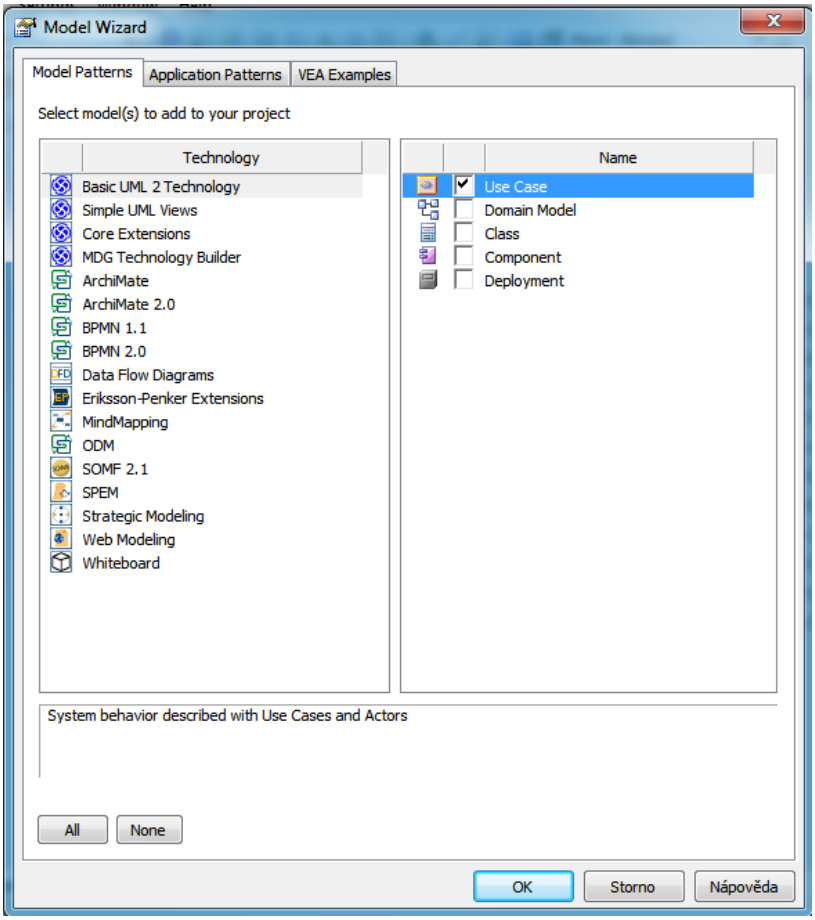
K modelování softwarových požadavků je možné též používat pokročilé počítačové systémy. V rámci práce byly vyzkoušeny programy Enterprise Architect ve zkušební verzi a MagicDraw Personal Edition testovací univerzitní licence, oba poskytovaly po omezenou dobu plnou funkcionalitu produktu.

Oba programy podporují všechny typy UML diagramů, zmíněné v předchozí kapitole, ke kterým je možné přidat další informace, například scénáře případů užití, nebo přímo kód vyvíjeného programu.

V obou programech je nejprve nutné založit projekt, a poté do něj vkládat vybrané diagramy, ovšem tato kapitola se zaměřuje pouze na UML diagramy užití.

### 3.2.1 Enterprise Architect

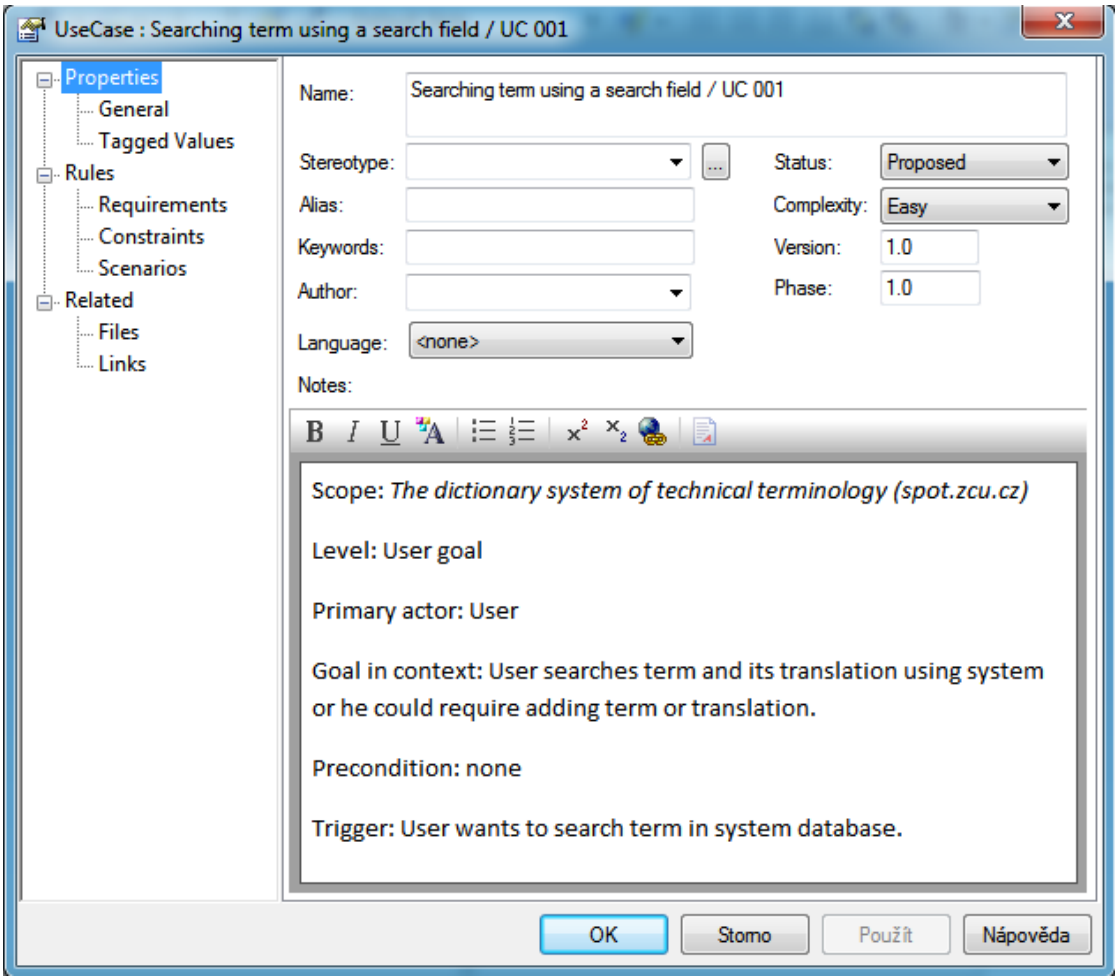
Enterprise Architekt je robustním nástrojem pro modelování a návrh UML diagramů.




*Obr. 3 Enterprise Architect – Volby při vytváření projektu*

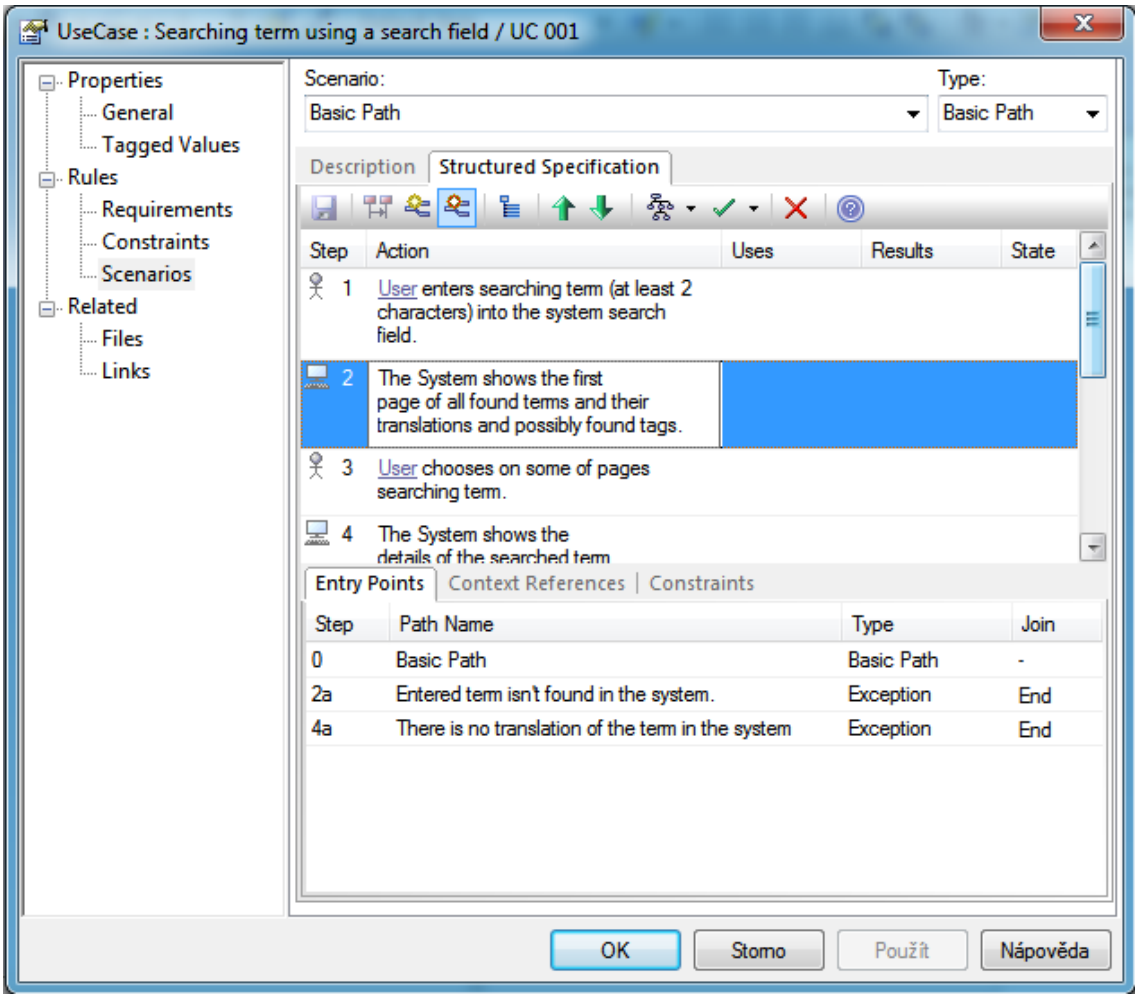
Při zakládání projektu v EA je možné zvolit i vytvoření UML diagramu užití (Use Case), viz obrázek 3. Po jeho vytvoření přidáme do diagramu (Primary use cases) jednotlivé aktéry a pojmenujeme je, a následně také případy užití. Poté je třeba vše propojit a zavést odpovídající vazby mezi aktéry a případy a případně i navzájem, pokud například používáme případy užití, které jsou rozšířením případů jiných. Je možné využít 2 předpřipravené případy užití a 1 aktéra, které se automaticky vytvoří při zvolení příslušné volby.

Po rozkliknutí případů užití je možné do programu přidat body scénáře, na ně navázat rozšíření a přidat další informace o případu. Bohužel EA nemá potřebné položky šablony A. Cockburna, takže je třeba je umístit nestrukturovaně do textového pole s poznámkami případu, viz obrázek 4.




*Obr. 4 EA – Vložení položek šablony do případu užití*

Scénáře je možné vkládat do EA jako souvislý text, nebo strukturovaně po jednotlivých bodech, kdy systém kroky sám čísluje a zvýrazňuje názvy aktérů, jak je vidět na obrázku 5.




*Obr. 5 EA – Tvorba strukturovaného scénáře*

V případě strukturovaného scénáře se rozšíření přidává výběrem odpovídajícího bodu standardního scénáře a následného zvolení tlačítka přidat alternativní cestu (Add Alternative Path) nebo přidat chybovou cestu (Add Exception Path), na obrázku 5 zvýrazněné tlačítko nahoře. Po zadání případů užití je možné vygenerovat celou řadu výstupů, ať už obrázek z diagramu nebo dokumentaci s kompletním zněním případů v různých formátech.

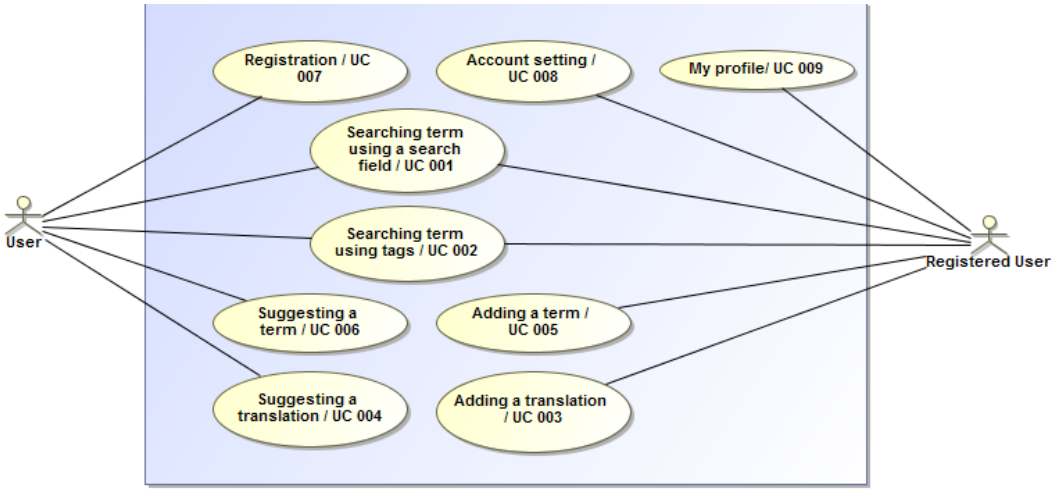
### 3.2.2 MagicDraw

MagicDraw je stejně jako EA program pro softwarovou analýzu.

MagicDraw umožňuje při prvním spuštění zvolit takzvanou perspektivu, která zpřístupní odpovídající úroveň funkcionality Tu je možno měnit i v průběhu práce s programem. Začínající uživatel si tak může nechat skrýt pokročilé funkce a snadněji se seznámit se systémem.

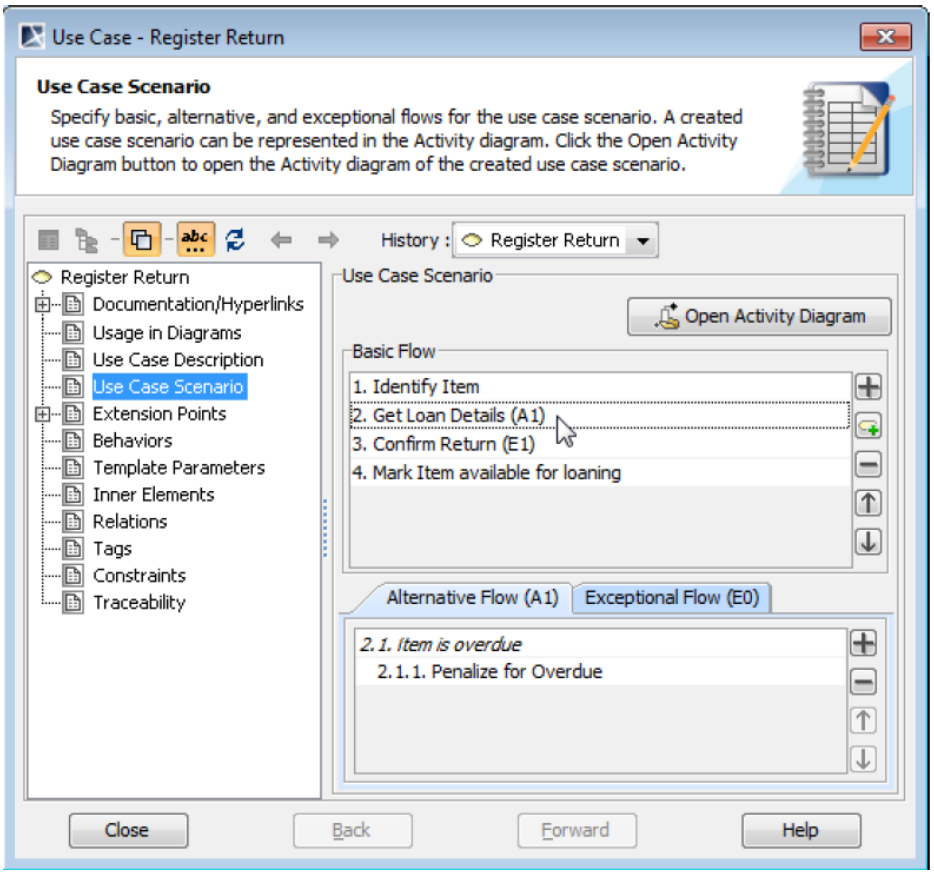
V programu MD je o něco obtížnější vytvořit diagram užití než v EA, kde se vytvoří s projektem, je třeba nalézt odpovídající ikonu diagramu užití (Use case diagram). Jak diagram v programu

vypadá, vidíte na obrázku 6.




*Obr. 6 MagicDraw - UML diagram systému SPOT*

Poté už se se systémem pracuje obdobně jako s Enterprise architektem, opět zadáváme jednotlivé kroky postupně do scénáře a přidáváme k nim alternativní a chybové cesty, viz obrázek 7.




*Obr. 7 MD – Práce se scénářem [No Magic Inc. 2014]*

# 4. Vlastní práce

## 4.1 Úvod

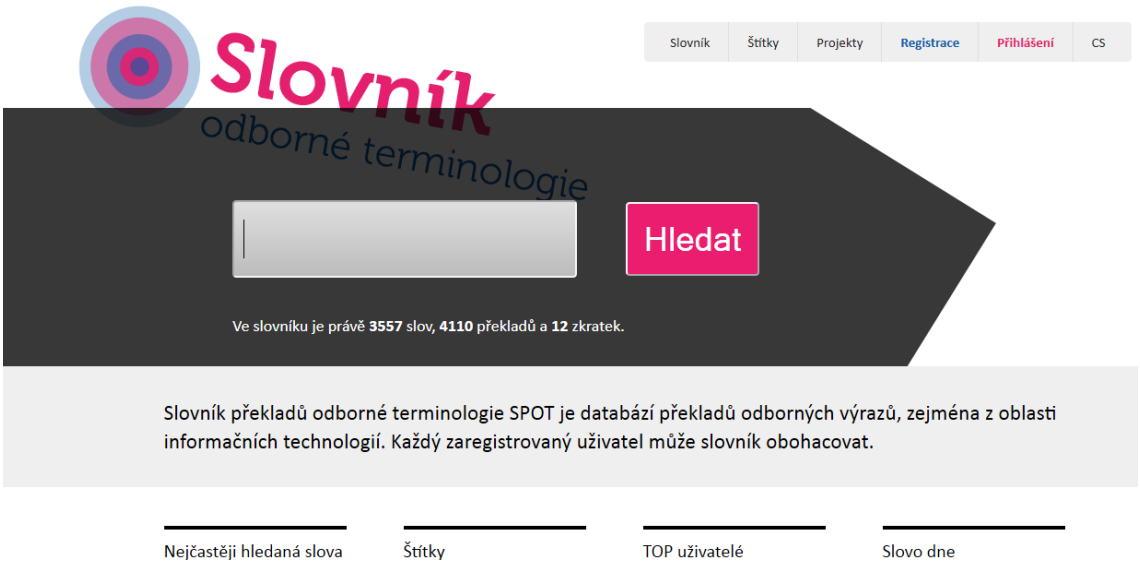
Tématem vlastní práce bylo seznámit se se základy softwarových požadavků a s tvorbou případů užití, viz předchozí kapitoly, a vytvořit případy užití pro již existující systém. Následně přibyla i úloha analyzování již hotových případů užití.

Kapitola vlastní práce popisuje postup analýzy a tvorby případů užití pro systém SPOT a podrobnou analýzu případů užití CoCoME. K obojímu je přihlíženo z hlediska doporučení Alistaira Cockburna, která uvádí ve svých knihách, a podle zkušeností autora práce. Tato kapitola by měla ukazovat, jak konkrétně se případy užití tvoří, a na čem všem závisí jejich přehlednost.

## 4.2 Vytváření případů užití odpovídajících pravidlům

V rámci práce byly vytvořeny případy užití systému SPOT podle pravidel daných Alistairem Cockburnem.

### 4.2.1 SPOT - Slovník překladů odborné terminologie




*Obr. 8 Snímek obrazovky systému SPOT*

Slovník překladů odborné terminologie (SPOT) je projektem Katedry informatiky a výpočetní techniky Západočeské univerzity v Plzni, který si klade za cíl vytvořit ucelený soubor překladů odborných termínů z oblasti informačních a komunikačních technologií.

V době psaní této práce slovník obsahoval přes 3500 slov, přes 4000 překladů ale jen pouhých 12 zkratek. S přibývající uživatelskou základnou však mohou tyto počty rychle narůstat, systém

je volně přístupný a kdokoliv do něj může navrhovat nová slova, zkratky i překlady. Po registraci do systému se otevírají další možnosti, jako jsou komentáře ke slovům, hodnocení překladů a podobně. Úspěch systému tak závisí na míře zapojení odborné veřejnosti do projektu a chuti uživatel přispívat do systému novými slovy K tomu jsou motivováni ziskem bodů a mohou se tak objevit na hlavní stránce v seznamu TOP uživatel. SPOT také umožňuje vložit si vyhledávací okénko do vlastních webových stránek.

### 4.2.2 Analýza aktérů

Před začátkem analýzy bylo třeba se seznámit se systémem, nejprve jako uživatel a posléze jako registrovaný návštěvník webu. Z toho a z nápovědy na webu nakonec vyšlo rozdělení aktérů případů užití na Uživatele, Registrovaného uživatele a Administrátora. Web ještě uvádí roli Moderátora, ale k ní uvádí stejná práva, která má Administrátor. Tyto dvě role nebylo možné vyzkoušet a zjistit mezi nimi rozdíly, takže role Moderátora není dále v práci uváděna.

Bylo by též možné použít jiné rozdělení aktérů, například na uživatele hledající a uživatele, kteří chtějí do slovní zásoby přispět, na odborníky a laiky a podobně. Pak by bylo ovšem z hlediska možností práce se systémem stále potřeba rozlišovat, zda je aktér registrován či nikoliv, což by bylo možné prostřednictvím položky Předpoklad (Precondition) v případu užití, ale snižovalo by to celkovou přehlednost práce. Navíc případy užití jako je například Registrace by vykonávali jak hledající, tak sdílející uživatelé. Proto bylo zvoleno právě rozdělení na registrované a neregistrované.

### 4.2.3 Výběr případů užití

Ze zjištěných aktérů a funkcí systému byl následně vytvořen UML diagram užití, který je přílohou práce. Poté byly po dohodě s vedoucím práce vybrány některé případy užití k dalšímu rozepsání. Těmi byly vybrány vyhledávání termínu jak pomocí vyhledávacího pole, tak pomocí štítků, registrace, nastavení účtu, můj profil a navržení a přidání termínu a navržení a přidání překladu.

Rozdíl mezi navržením a přidáním je ten, že neregistrovaný uživatel smí slovo pouze navrhnout, zatímco registrovaný může slovo do databáze přidat. Navržená slova jsou zařazena do seznamu přání (Wishlist). Registrovaným uživatelem přidané slovo se však do schválení administrátorem zobrazuje pouze přihlášeným uživatelům.

Vyhledávání pomocí vyhledávacího pole využívá pole na hlavní stránce projektu, viz obr. 8. U vyhledávání pomocí štítků je nejprve na stránce Štítky zvolen odpovídající a poté na stránkovaném seznamu termínů označených daným štítkem ručně vybraný hledaný termín.

Případy užití Nastavení účtu a Můj profil mohou pochopitelně vykonávat pouze zaregistrovaní. Postup registrace popisuje daný případ užití.

### 4.2.4 Rozsah

Všechny případy užití systému mají Rozsah (Scope) stanoven celý systém SPOT a na systém je políčeno jako na černou skříňku, vzhledem k tomu, že byl analyzován bez možnosti přístupu k jednotlivým komponentám systému a vzhledem k domluvě se zadavatelem práce.

| Scope: *The dictionary system of technical terminology (spot.zcu.cz)* |
|---|

To, že je k systému přistupováno ako k černé skříňce, není v případech užití nijak označeno, protože šablona A. Cockburna neobsahuje žádnou odpovídající položku. Ovšem autor doporučuje použití symbolu černé skříňky vedle názvu případu užití [Cockburn 2001] Ta také nebyla použita, protože účelem případů je jejich další strojové zpracování.

### 4.2.5 Úroveň

Všechny případy užití mají Úroveň (Level), Uživatelský cíl, nejsou si nijak vzájemně nadřazeny či podřízeny, proto mají všechny stejnou úroveň. Uživatelským cílem na úrovni Podfunkce (Subfunctions) by bylo například přihlášení do systému, které je třeba před zahájením případů užití registrovaných uživatelů, ale to bylo pro svou přílišnou krátkost hlavního scénáře vyřazeno.

| Level: User goal |
|---|

Případ užití Registrace nemá úroveň podfunkce proto, že při aplikování Cockburnovy rady o možnosti odejít na kávu ihned po skončení případu můžeme konstatovat, že je možné odejít, protože mezi samotnou registrací na webu a obdržením aktivačního e-mailu je časová prodleva, po které již uživatel nemusí mít pro danou chvíli zájem se systémem dále pracovat a může ihned po aktivaci systém opustit a dojít si například na kávu. Na rozdíl od přihlášení, kde si lze jen obtížně představit, že by se člověk úspěšně přihlásil do systému a ihned poté systém opustil.

### 4.2.6 Aktéři

Aktérem (actor) v případech užití systému SPOT je vždy uživatel, ať už registrovaný či nikoli. Mezi případy užití není žádný, který by obsahoval sekundární aktéry, či kde by primárním nebo sekundárním aktérem bylo nějaké softwarové nebo hardwarové vybavení. To je dáno jednak

pohledem na systém jako na černou skříňku, a pak také jednoduchostí případů užití, kde do scénáře nezasahuje žádná další osoba.

| Primary actor: Registered User |
|---|

Příkladem případu užití, kde by se objevil sekundární aktér, by například bylo přidání termínu do databáze, pokud bychom chtěli zachytit celý proces přidání do veřejné části databáze systému, kde je nutné, aby termín schválil administrátor. Administrátor by v takovém případě byl sekundárním aktérem a uživatel primárním.

### 4.2.7 Kontext cíle, Předpoklady, Spouštěče a Následné podmínky

Kontext cíle (Goal in context) je ve všech případechpsán z pohledu primárního aktéra a cíle jeho počínání.

Předpokladem (Precondition) je v případech užití umístění uživatele na konkrétní stránce, znalost překladu termínu nebo jeho stav (přihlášení).

Spouštěčem je uživatelovo přání a u případů užití týkajících se přidávání, či návrhů termínů a překladů je spouštěčem také nenalezení daného termínu nebo vhodného překladu.

Následnou podmínkou (Postcondition) je ve všech případech, kde se nachází, uložení získaných dat do systémové databáze

### 4.2.8 Hlavní scénář a Rozšíření

Hlavní scénář (Main success scenario) je psán vždy stylem Uživatel – Systém tak, aby bylo vždy jasné, kdo je „na tahu“. Scénář je psán jednoduchým jazykem, jak je doporučováno. Hlavní scénář po konzultaci s vedoucím práce obsahuje také názvy položek formulářů, které systém obsahuje tak, aby se teoreticky dal sestavit podle případů užití.

Main success scenario:

1. **User** chooses tag where is according to him searching term.
2. **The system** shows first page of the list of tagged terms and their translations.
3. **User** chooses on some of pages a term.
4. **The system** shows the page with translations of searching term and the term details to the user.
5. **User** chooses the most appropriate one from all offered translations.

*Příklad interakce aktér - systém*

Rozšíření (Extensions) obsahuje obvykle chybové scénáře. Jen u případu užití Můj profil jde skutečně o rozšíření a nikoli zachycení chyb.

Extensions:

2a User could click on My added words, My added abbreviations and My added translations to see more terms.

2a1 System shows list off all corresponding terms a translations a their status.

2b User could click on My comments.

2b1 System shows list off all user's comments and terms to which they relate.

*Příklad rozšíření v případu užití můj profil, které není zachycením chyby.*

Původně všechny případy užití obsahovaly rozšíření, které zachycovalo selhání systému v jakémkoliv bodě s následným ukončením případu, to však bylo vyřazeno pro nadbytečnost.

### 4.2.9 Případy užití

Případy užití byly kvůli požadavku strojového zpracování vypracovány v angličtině a všechny jsou přílohou práce.

Případ užití Registrace byl několikrát zmíněn v předchozích odstavcích, proto ho zde uvádím jako příklad případu užití.

### *Registration / UC 007*

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: User

Goal in context: User signs up into the system database to have access to more system functions.

Precondition: User is on the registration page.

Trigger: User wants to sign up into the spot system.

Postcondition: User has been added to the database and the account has been activated.

Main success scenario:

1. User fills in the registration form
   (Login, Password, again password, E-mail and determines if the e-mail is public or not and enters the verification text.
2. System notifies user that the form was send and prompts to activation per e-mail.
3. User activates account using link in the received e-mail.
4. System notifies that the account is activated.

Extensions:

1a. User login is already in the system.

1a1 System notifies it to the user.

1a2 User enters other login.

1b Passwords do not match

1b1 System notifies that.

1b2 User corrects an error.

1c The verification text is incorrectly filled.

1c1 System notifies it and generates a new one.

1c2 User rewrites a new verification text.

2a System couldn’t send the activation mail

2a1 System notifies it and prompts user to register later.

3a User hasn't activated an account.

3a System informs the user that his account isn’t activated while he is trying to log in.

### 4.2.10 Zkušenosti ze psaní případů užití systému SPOT

Zpočátku se zdálo snadné vytvořit hlavní scénáře pro již existující systém, avšak ukázalo se, že je to často obtížnější, než tvořit případy užití pro teprve vznikající projekt. U teprve vznikajícího softwaru je totiž nutné zachytit všechny detaily, které má daný systém obsahovat, jinak v něm prostě obsaženy nebudou. Na druhou stranu pokud se něco přehlédne v již hotovém řešení, nemusí si toho téměř nikdo všimnout. Z toho důvodu bych jasně doporučil tvořit případy užití raději předem, než ex-post, ty totiž nemusí zachycovat skutečný a přesný stav dokumentovaného systému nebo je to přinejmenším obtížněji zachytitelné.

Při dokumentaci již vytvořeného systému je však snazší zachycení chybových situací, protože si člověk může systém vyzkoušet a navíc pokud je systém ve fázi krátkého nasazení do ostrého

provozu, na některé chyby analytik při zkoušení sám narazí, není tak třeba tolik fantazie, jako při dokumentování zatím neexistujícího softwaru.

Při pročítání jakýchkoli případů užití by měl mít člověk pocit, že je jejich tvorba snadná a jednoduchá, avšak není to tak úplně pravda Samotného mě mnohokrát na konzultaci překvapilo, kolik chyb se může v případu užití vyskytnout. Je třeba mít stále na paměti, že je nutné zachytit vše a všechny situace, které mohou nastat v průběhu scénáře.

Šablonu od A. Cockburna hodnotím jako velice zdařilou, která obsahuje všechna potřebná data vhodně strukturovaná. Je jasné, proč právě s ní pracuje poloautomatické strojové zpracování případů užití.

## 4.3 Analýza existujících specifikací případu užití

Další částí práce byla analýza případů užití, které byly vytvořeny pracovníky jedné italské a německých univerzit.

### 4.3.1 CoCoME - The Common Component Modeling Example

Projekt CoCoME – The Common Component Modeling Example [Rausch 2008] neboli příklad modelování komponent, je ukázkou aplikace, která poskytuje případy užití k výzkumu softwarového modelování [Bulej 2014]. Případy užití jsou v angličtině.

Případy užití popisují obchodovací systém fiktivní sítě obchodních domů, zajišťující prodej, který je popsán od vybavení jednotlivých pokladen a práce samotného pokladního přes server prodejny až po centrální server, který je připojen ke všem serverům prodejen a poskytuje řadu údajů manažerovi prodejen. Autoři sami v dokumentu přiznávají, že vzhledem k tomu, že data byla získána od obchodní společnosti, můhou být neúplná nebo nepřesná [Rausch 2008].

### 4.3.2 Postup analýzy

Případy užití používají jinou šablonu než A. Cockburn, a tím pádem se liší od předchozích případů užití systému SPOT. Prvním bodem postupu analýzy daných případů užití bylo projít k nim přiložený popis celého systému, hardwaru a UML diagram užití systému, který naleznete v příloze. Následoval průchod případy užití s cílem zachytit rozdíly mezi šablonou použitou týmem CoCoME a tou od A. Cockburna, a až poté analýza jednotlivých případů užití. Z toho vznikla myšlenková mapa (mind map), která je také přílohou práce a byla použita při tvorbě následující analýzy.

### 4.3.3 Rozdíly od šablony případů užití podle Alisteira Cockburna

| | |
|---|---|
| Scope: | Brief Description. |
| Level: | |
| Primary actor: | Involved Actors. |
| Goal in context: | |
| Precondition: | Precondition. |
| Trigger: | Trigger. |
| Postcondition: | Postcondition. |
| Main success scenario: | Standard Process |
| Extensions: | Alternative or Exceptional Processes |

*Šablony A. Cockburna (vlevo) [Cockburn 2001] a šablona používaná CoCoME (vpravo) [Rausch 2008].*

Všechny případy užití CoCoME obsahují položku Brief Description, čili stručný popis daného případu, tuto položku Cockburnova šablona neobsahuje, nicméně lze ji přirovnat k popsané položce Kontext cíle (Goal in context), kterou CoCoME nepoužívá, i když se jejich obsah nemusí přímo shodovat.

Šablona použitá CoCoME dále nerozlišuje mezi položkami primární a sekundární aktér a všechny aktéry vypisuje v rámci jediné položky Zúčastnění aktéři (Involved Actors), kam tím pádem spadá také hardware.

Šablona obsahuje Předpoklady (Preconditions), Spouštěč (Trigger) a Následné podmínky (Postconditions). Hlavní scénář nazývá Standardní proces (Standard Process), což je jen otázka názvosloví. Standardní proces neobsahuje vždy interakce aktér-systém, po kroku aktéra není následující krok věnován systému.

Ve standardním scénáři se také nacházejí v hranatých závorkách kódy, které odkazují na takzvané zvláštní funkční vlastnosti (Extra-Functional Properties), například o kódu [p13-1], který se vztahuje k načítání položek do pokladny, v tabulce požadavků najdeme, že pravděpodobnost načtení čtečkou čárových kódů by měla být 99%. První písmena kódu udávají

typ požadavku (čas, číslo, míra příchodů), následuje číslo shodující se s číslem případu užití a číslo kroku ve scénáři, kde je požadavek použit, a pořadové číslo požadavku v rámci kroku [Rausch 2008].

Rozšíření (Extensions) se v šabloně nazývá Alternativní nebo chybové procesy (Alternative or Exceptional Processes). Každý podbod Alternativního procesu je uvozen slovy „V bodě (In step) X“ a poté posláno, co se v daném bodě scénáře stane jinak, což je rozdíl oproti Cockburnovým body X.a, X.b atd.

Šablona neobsahuje Rozsah (Scope), Úroveň (Level) a už zmíněný Kontext cíle (Goal in context).

Proč autoři za názvy položek píší tečku, se nepodařilo zjistit, Alistair Cockburn používá dvojtečku, která se jeví jako výrazně logičtější. Nebezpečí, že by se v obsahu položky objevila dvojtečka a snížila tak přehlednost, je výrazně nižší než u tečky.

### 4.3.4 Zhodnocení rozdílů šablon

Položku stručný popis (Brief Description) osobně hodnotím jako vhodnou, jde o shrnutí celého případu užití. To ale neznamená, že by, dle mého názoru, mohla nebo měla nahradit Cockburnův Kontext cíle, který je důležitý k popsání cíle a širších souvislostí v systému.

Nerozlišování šablony mezi primárním a sekundárními aktéry není příliš vhodné a snižuje přehlednost celých případů užití. Bylo by vhodné primárního aktéra alespoň nějak zvýraznit. Nazývat hlavní úspěšný scénář standardním procesem je v pořádku, i když slovo scénář je asi významově přeci jen o něco bližší, jde zde ale o jazykové preference.

Absence stálé interakce aktér-systém umožňuje některé standardní procesy případů užití významně zkrátit, avšak neumožňuje přesně popsat, jak má na danou akci systém reagovat, například co má přesně zobrazit aktérovi na monitoru po akci aktéra (např. stisknutí tlačítka aktérem). To by mělo, pokud je to pro projekt důležité, být v případu užití vždy uvedeno. Interakci lze vypustit jen tam, kde je naprosto bez pochybností jasné, jak se má systém na daný příkaz aktéra zachovat nebo tam, kde se takové chování (uživatelské rozhraní) definuje mimo případy užití.

Zvolený systém specifikování dalších požadavků, v tomto případě spíše zavádění měřitelných systémových parametrů, nazývaných zvláštní funkční vlastnosti, a jejich názvy (zkratky) jsou velice dobrým nápadem, a pokud má čtenář při čtení případů užití k dispozici zároveň i seznam těchto parametrů, může být orientace v nich velice snadná a jednoduchá. Jde o jednoduchou

pomůcku, s jejíž pomocí lze v relaci na kroky scénáře definovat další vlastnosti a požadavky na daný systém.

Název rozšíření „Alternativní nebo chybové procesy“ (Alternative or Exceptional Processes) je více vypovídající, avšak ideální by bylo držet se Hlavního scénáře a případná rozšíření pojmenovat jako Alternativní nebo chybové scénáře, ale to už je zmíněná otázka jazykového citu.

Zahájení alternativního průběhu slovy „V bodě“ (In step) je přesnější a výrazně pochopitelnější, než pouhé přidání písmene za číslo alternovaného kroku standardního procesu To lze hodnotit jako dobré zlepšení orientace v rozšířeních.

Rozsahem případu užití obvykle bývá celý diskutovaný systém, takže vypuštění této položky ze šablony, pokud jsou jasně dané a neměnné hranice projektu pro všechny případy užití, nezpůsobí žádné problémy s porozuměním textu, což je případ právě CoCoME. U rozsáhlejších systémů nebo tam, kde je třeba systém dokumentovat po jednotlivých komponentách, však nelze vypuštění Rozsahu doporučit.

S Úrovní je situace podobná, ta má za úkol rozlišit hierarchické vztahy mezi jednotlivými případy užití, a pokud jsou všechny případy užití na stejné úrovni nebo jsou snadno pochopitelné, lze tuto položku také vyřadit. Opět ale v případě složitějších systémů s případy užití rozdílných úrovní odebrání této položky pouze zkomplikuje orientaci v dokumentaci.

Celkové zhodnocení případů užití naleznete za analyzovanými případy užití.

### 4.3.5 Analýza jednotlivých případů užití CoCoME - The Common Component Modeling Example

Následuje analýza jednotlivých případů užití CoCoME, před kterou jsou vždy uvedeny části jednotlivých případů, důležité pro následnou analýzu, celé případy užití jsou v příloze práce. Analýza je rozdělena na jednotlivé případy užití, které jsou hodnoceny jak podle doporučení A. Cockburna, tak podle osobního názoru autora práce z věcného i formálního hlediska.

### *UC 1 – Proces sale [Rausch 2008]*

***…***

***Involved Actors.*** Customer, Cashier, Bank, Printer, Card Reader, Cash Box, Bar Code Scanner, Light Display

***Trigger.*** Coming to the Cash Desk a Costumer wants to pay his chosen product items.

***Standard Process***

1. The Customer arrives at the Cash Desk with goods to purchase. [arr1]

2. The Cashier starts a new sale by pressing the button Start New Sale at the Cash Box. [t12-1]

3. The Cashier enters the item identifier. This can be done manually by using the keyboard of the Cash Box [p13-1, t13-1] or by using the Bar Code Scanner[p13-2, t13-2].

4. Using the item identifier the System presents the corresponding product description, price, and running total. [t14-1]

***The steps 3-4 are repeated until all items are registered.*** *[n11-2]*

5. Denoting the end of entering items the Cashier presses the button Sale Finished at the Cash Box. [t15-1]

**(a)** To initiate **cash payment** the Cashier presses the button Cash Payment at the Cash Box. [p15-1,t15a-1]

i. The Customer hands over the money for payment. [t15a1-1]

ii. The Cashier enters the received cash using the Cash Box and confirms this by pressing Enter. [t15a2-1]

iii. The Cash Box opens. [t15a3-1]

iv. The received money and the change amount are displayed [t15a4-1], and the Cashier hands over the change. [t15a4-2]

v. The Cashier closes the Cash Box. [t15a5-1]

**(b)** In order to initiate **card payment** the Cashier presses the button Card Payment at the Cash Box. [p15-2, t15b-1]

i. The Cashier receives the credit card from the Customer [t15b1-1] and pulls it through the Card Reader. [t15b1-2]

ii. The Customer enters his PIN using the keyboard of the card reader and waits for validation. [t15b2-1]

***The step 5.b.ii is repeated until a successful validation or the Cashier* presses the button for cash payment.** [t15b2-2, n15b2-1]

…

***Alternative or Exceptional Processes***

– In step 3: Invalid item identifier if the system cannot find it in the Inventory. [p13-4]

1. The System signals error and rejects this entry. [t13-3]

2. The Cashier can respond to the error as follows:

(a) It exists a human-readable item identifier: [p13-5]

i. The Cashier manually enters the item identifier. [t13-4]

ii. The System displays the description and price. [t14-1]

(b) Otherwise the product item is rejected. [p13-6]

**– In step 5.b: Card validation fails. [p15b2-2]**

**1. The Cashier and the Customer try again and again.**

**2. Otherwise the Cashier requires the Customer to pay cash.**

– **In step 6: Inventory not available. [p16-1]**

**The System caches each sale and writes them into the Inventory as soon as it is available again.** [t161-1]

V use casu číslo 1 – Proces prodeje - je vidět již zmiňovaný velký počet Zúčastněných aktérů (Involved Actors), který nepůsobí příliš přehledně, navíc na prvním místě je uveden zákazník (Customer) i přesto, že primární aktérem by byl pokladní (Cashier), který se systémem přímo pracuje, čili minimálně by bylo vhodné pro větší přehlednost změnit pořadí, pokud budeme zachovávat šablonu.

U aktéra typu Zákazník ještě zůstaneme. Hned v prvním kroku totiž zahajuje případ užití příchodem ke kase, což je ale také spouštěč celého případu užití Tento bod je tak ve scénáři zbytečný.

Po bodu 4 standardního procesu následuje poznámka o možné opakovatelnosti bodů 3 a 4, což je určitě výrazně přehlednější, než kdyby se autoři pokoušeli tento fakt zachytit uvnitř popisu alternativního procesu, a stejnou možnost nabízí i Cockburn.

Hlavní scénář, jak je vidět z ukázky, obsahuje dvě alternativní cesty v následujícím bodě číslo 5, těmi je platba hotově (a) cash payment), nebo platba kartou (b) card payment). To není příliš běžné v rámci případů užití. Hlavní scénář by měl obsahovat nejkratší a nejjednodušší průchod systémem [Cockburn 2001] Těžko však určit, která z možností to je. Platba kartou má nižší počet kroků, ale může při ní nastat více chyb (špatně zadaný PIN, ověření platby atd.), které pokud nejsou vyřešeny, vedou k platbě v hotovosti. Lze tak říci, že zvolené řešení, kdy jsou obě

situace vyrovnané, je vhodné a celkově přispívá k přehlednosti celé situace. Není tak třeba hledat jednu z možností v Alternativních procesech.

Po bodu 5.ii následuje opět poznámka, že bod je možné opakovat, nebo zvolit platbu hotově a stejnou informaci obsahuje také kapitola Alternativní nebo chybové procesy. Předchozí poznámku o možnosti opakování kroků 3 a 4 však neobsahuje, což je chyba, protože by případ užití měl být konzistentní. Buď je třeba vyřadit tuto poznámku z chybových scénářů, což by byla sama o sobě chyba, nebo raději do těchto scénářů zařadit i poznámku o bodech 3 a 4, popsanou jako požadavek na přidání další položek nákupu při bodě 5.

Krok řešení chybového stavu týkajícího se bodu 6 – „zásoby nejsou k dispozici“ není očíslovaný a jde zřejmě o záměr, který se objevuje i u dalších případů užití Jedná se tak ale o nerozepsání řešení do jednotlivých kroků, což však tady není třeba, zde opravdu stačí bod očíslovat.

### *UC 2 – Správa expresní pokladny[Rausch 2008]*

***Involved Actors.*** Cashier, Cash Box, Light Display, Card Reader

***Precondition.*** The Cash Desk is either in normal mode and the latest sale was finished (case 1) or the Cash Desk is in express mode (case 2).

***Trigger.*** **This use case is triggered by the system itself.**

***Postcondition.*** The Cash Desk has been switched into express mode or normal mode. The Light Display has changed its color accordingly.

***Standard Process***

**1.** The considered Cash Desk is in normal mode [p2-1] and just finished a sale which matches the condition of an express checkout sale. Now 50% of all sales during the last 60 minutes fulfill the condition for an express checkout.

…

2. The Cash Desk is in express mode [p2-2] and the Cashier decides to change back into normal mode.

…

Případ užití číslo 2 pojednává o přepínání poklady mezi expresním a normálním módem a je rozšířením případu užití číslo 1 právě tím, že přidává tuto funkcionalitu pokladně. V expresním módu pokladna přijme nákup o maximálně 8 položkách a nepřijímá platbu kartou, je však možné přepnout zpět na normální mód a obojí umožnit.

Nepovažuji za vhodné toto specifikovat právě prostřednictvím zvláštního případu užití. Je možné tuto věc přidat do předchozího případu tím, že by byl napsán pro situaci pokladny v expresním módu, což je nejrychlejší a nejjednodušší průchod a rozšířením by bylo popsáno přepnutí pokladny a akceptace platby kartou.

I tento případ užití obsahuje delší seznam zúčastněných aktérů, avšak pokladní je správně na prvním místě a po něm následuje už jen hardware. Spouštěč případu hovoří o tom, že případ spouští systém sám. To je sice možné, ale není to dostatečně vypovídající. Měl by raději říkat, kdy bude případ zahájen než jen kým. Předpoklad by tak měl být spíše spouštěčem případu.

### *UC 3 – Objednávka produktů [Rausch 2008]*

***Involved Actors.*** Store Manager

…

***Postcondition.*** The order was placed and a generated order identifier was presented to the Store Manager.

***Standard Process***

**1.** A list with all products [n3-1] and a list with products running out of stock are shown. [n3-2, p3-1, t31-1]

2. The Store Manager chooses the product items to order and enters the corresponding amount. [t32-1]

3. The Store Manager presses the button Order at the Store Client's GUI. [t33-1]

**4.** The appropriate suppliers are chosen and orders for each supplier are placed. An order identifier is generated for each order and is shown to the Store Manager. [t34-1, t34-2, t34-3]

Tento případ užití řeší objednávky produktů manažerem prodejny, ten je zde uveden jako jediný aktér. Případ užití začíná ve Standardním procesu informací, že jsou zobrazeny seznamy produktů, chybí však informace kým. Mělo by zde být uvedeno, že systém zobrazí seznamy, nikoli „seznamy jsou zobrazeny".

Dále případ užití obsahuje zbytečně oddělený krok potvrzení celé objednávky tlačítkem, který by mohl být součástí bodu předchozího. Po odeslání objednávky stisknutím tlačítka by tuto skutečnost měl systém oznámit. Poslední 4. bod je spíše následnou podmínkou, posledním bodem by mělo být pouze potvrzení objednávky systémem a zobrazení identifikátorů objednávek. Řeč o tom, že jsou vybráni dodavatelé a objednávky zařazeny do systému, by

spadala do následných podmínek, ze kterých by bylo možné smazat větu o zobrazení identifikátorů objednávek, která by zůstala součástí posledního bodu.

Případ užití neřeší chyby, které mohou nastat, například při zvolení většího než dostupného množství výrobků manažerem prodejny, s problémy při odeslání objednávky a při volbě vhodného dodavatele systémem.

### *UC 4 – Přijetí objednaného zboží [Rausch 2008]*

***Brief Description.*** Ordered products which arrive at the Store have to be checker for correctness and inventoried.

***Involved Actors.*** Stock Manager

***Precondition.*** The Store Client was started and the part Inventory of the Trading System is available.

***Trigger.*** The ordered products arrive at the Store.

***Postcondition***. The Inventory is updated with the ordered products.

***Standard Process***

**1.** Ordered products arrive at the stock attached by an order identifier which has been assigned during the ordering process. [n4-1]

2. The Stock Manager checks the delivery for completeness and correctness. [p4-1, t42-1]

3. In the case of correctness, the Stock Manager enters the order identifier and presses the button Roll in received order. [t43-1]

**4.** The Trading System updates the Inventory. [t44-1]

***Alternative or Exceptional Processes***

– **In step 2:** Delivery not complete or not correct. [p4-2]

The products are sent back to the supplier and the Stock Manager has to wait until a correct and complete delivery has arrived. This action does not recognized by the System.

4. případ užití popisuje příjem objednaného zboží manažerem prodejny a je zde jako jediný umístěn celý, protože o něm ještě bude řeč dále.

V případ užití je opět chybný první krok scénáře, ten by měl být spouštěčem celého případu a naopak poslední bod je totožný s následnou podmínkou. Druhý a třetí krok lze sjednotit, protože systém reaguje až na vložení a potvrzení dat. Posledním krokem by opět mělo být potvrzení zpracování přijaté objednávky.

Řešení chybového procesu by bylo opět vhodné očíslovat a uvést aktéra, který daný krok vykoná, a uvést pouze vrácení zboží zpět k dodavateli („1. Manažer prodejny odešle zboží zpět.“). Novým příchodem zbožím totiž začíná případ užití znovu. Chybový proces by měl také řešit, co se děje poté, co dorazí nekompletní nebo nesprávná dodávka zboží z hlediska dalšího dočasného fungování prodejny.

### *UC 5 – Zobrazení skladových zásob*

***Involved Actors.*** *Store Manager*

…

***Postcondition.*** The report for the Store has been generated and is displayed on the reporting GUI.

***Standard Process***

1. The Store Manager enters the store identifier and presses the button Create Report. [t51-1]

**2.** A report including all available stock items in the store is displayed. [t52-1]

Případ užití Zobrazení skladových zásob je velice krátký, má jen 2 body scénáře. Navíc se opět poslední krok shoduje s Následnou podmínkou, kterou je tak možné vypustit (uvést žádné (none)).

### *UC6 – Zobrazení zpráv o doručení*

***Involved Actors.*** Enterprise Manager

…

***Postcondition***. The report for the Enterprise has been generated and is displayed to the Enterprise Manager.

***Standard Process***

1. The Enterprise Manager enters the enterprise identifier and presses the button

Create Report. [t61-1]

**2.** A report which informs about the mean times is generated. [t62-1]

Stejně jako předchozí je tento případ užití krátký a stejně tak duplikuje informace uvnitř Následné podmínky a posledního bodu procesu. Navíc jsou si oba případy užití velice podobné, takže by bylo jistě možné napsat jeden případ o zobrazení výkazů pro aktéra „manažer“ a do

alternativ uvést, jaké zprávy si může vygenerovat a zobrazit který manažer. Oběma případům chybí zachycení chybových procesů (nedostupnost serveru, zapomenutí identifikátoru).

### *UC7 – Změna ceny*

…

***Postcondition****.* The price for the considered product has been changed and it will be sold with the new price now.

***Standard Process***

1. The System presents an overview over all available products in the store. [t71-1]

2. The Store Manager selects a product item [t72-1] and changes its sales price. [t72-2]

**3.** The Store Manager commits the change by **pressing ENTER.** [t73-1]

Změna ceny je další kratší případ užití, který působí dojmem, že se ho autoři snažili prodloužit přidáním bodu 3, který se týká potvrzení změny stisknutím klávesy ENTER. Místo toho by však měl opět obsahovat potvrzení systému o provedené změně ceny. Následná podmínka je u tohoto případu správně.

### *UC8 – Výměna produktů mezi prodejnami (při nízkém stavu zásob)*

***Brief Description.*** If a store runs out of a certain product (or a set of products; "required good"), it is possible to start a query to check whether those products are available at other Stores of the Enterprise ("providing Stores"). Therefore the Enterprise Server and the Store Servers need to synchronize their data on demand (one scheduled update per day or per hour is not sufficient). After a successful query the critical product can be shipped from one to other Stores. But it has to be decided (using heuristics to compute the future selling frequency), whether the transportation is meaningful. For example, if the product is propably sold out at all Stores within the same day, a transportation does not make sense.

Expressed in a more technical way one Store Server is able to start a query at the Enterprise Server. The Enterprise Server in turn starts a query for products available at other Stores. As the Enterprise Server does not have the current global data for Stores at any time (due to a write caching latency at the Store Servers) the Enterprise Server has to trigger all Store Servers to push their local data to the Enterprise Server.

***Involved Actors.*** This use case is not an end-user use case. Only servers are involved.

***Precondition.*** The Store Server with the shortage product is able to connect to the Enterprise Server.

***Trigger.*** This use case is triggered by the system itself.

***Postcondition.*** The products to deliver are marked as incoming or unavailable, respectively, in the according Stores.

***Standard Process***

**1.** A certain product of the Store runs out.

**2.** The Store Server recognizes low stock of the product. [t82-1]

3. The Store Server sends a request to the Enterprise Server (including an identification of the shortage products, and a Store id) [t83-1]

…

8. The Store Server of a near by Store is provided with information that it has to deliver the product. [t88-1]

**(a)** The required product is marked as unavailable in the Store. [t88-2]

***Alternative or Exceptional Processes***

**–** The Enterprise Server is not available: The request is queued until the Enterprise Server is available and then is send again.

**–** One or more Store Servers are not available: The Enterprise Server queues the requests for the Store Servers until they are available and then resend them. If a Store Server is not available for more than 15 minutes the request for this Server is canceled. It is assumed, that finally unavailable Store Servers do not have the required product.

Jak je vidět z ukázky části případu užití číslo 8, stručný popis (Brief Description) není zrovna krátký a stručný a jde spíše o vysvětlení stavu než o popis případu. Definice zúčastněných aktérů také není nijak ideální, je zde pouze konstatováno, že nejde o případ užití s konečným uživatelem, že se ho účastní pouze servery. Mělo by v této kolonce tak být napsáno Centrální server (Enterprise server) a Servery prodejen (Store servers) a poznámka, kterou aktuálně položka obsahuje, by mohla následovat po těchto údajích. Spouštěč, stejně jako v případě UC2, hovoří o tom, že se případ užití spouští sám, opět tak zde chybí, kým je zahájen a kdy.

První 2 kroky standardního procesu jsou vhodné stát se spouštěčem a proces by tak začínal až bodem 3. Následná podmínka se opět shoduje s bodem 8a, z toho důvodu je možné tento bod vypustit. Alternativní nebo chybové procesy opět nejsou očíslovány a strukturovány a jejich řešení není na rozdíl od ostatních případů odřádkováno od nastalé chyby. Stejně tak úplně chybí, k jakému bodu standardního procesu se vztahují. Dostupnost centrálního serveru by měla být předpokladem případu.

### *„Rozšíření“ UC8 – Odstranění stavu „na cestě“*

***Brief Description.*** If the first part of use case 8 (as described above) has passed, for moved products an amount marked as incoming remains at the Inventory of the Store receiving the products. An extension allows to change that incoming mark via a user interface at the Store Client if the moved products arrive at a Store.

***Precondition.*** The Inventory is available and the Store Client has been started.

***Trigger.*** The moved products (according to UC8) arrive at the Store.

***Postcondition***. For the amount of incoming products the status ”incoming” is removed in the Inventory.

***Standard Process***

**1.** The products arrive at the stock of the Store.

2. For all arriving products the Stock Manager counts the incoming amount.

3. For every arriving product the Stock Manager enters the identifier and its amount into the Store Client.

4. The system updates the Inventory.

***Alternative or Exceptional Processes***

– If the entered amount of an incoming product is larger than the amount accounted in the Inventory, the input is rejected. The incoming amount has to be re-entered.

Případ užití označený jako rozšíření případu předchozího řeší příjem zboží získaného výměnou mezi prodejnami a není zakreslen v UML diagramu projektu. Příjem zboží v systému už ale

dokumentuje případ užití číslo 4 – Přijetí objednaného zboží, tento případ užití je tak duplicitní a jeho jediná odchylka, která odstraní z databáze prodejen informaci, že je produkt na cestě, (incoming) by měla být přidána do případu 4.

Případy užití spolu ale nekorespondují v chybových procesech, kde případ číslo 4 řeší navíc nekompletnost a nesprávnost objednávky a „Rozšíření UC8“ řeší zadání většího než možného množství přijatých výrobků, obojí by tam mělo být zařazeno do upraveného případu užití číslo 4. V případě, že se nekontrolují i výrobky předávané mezi prodejnami, ale pouze se počítá jejich množství, muselo by být toto označeno v případu.

Informace bodu 1 standardního procesu je opět duplicitní se spouštěčem a měl by být vyřazen, pokud bychom z nějakého důvodu tento případ užití chtěli zachovat (například při jeho zpřesnění a zjištění větších množství odchylek od standardního příjmu zboží).

### 4.3.6 Celkové hodnocení případů užití CoCoMe

Celkově lze hodnotit případy užití označované jako CoCoME jako rozporuplné s řadou chyb. Přínosem šablony je položka stručný popis (Brief Description), kterou ale autoři v případě UC 8 špatně používají. Možnost alternativní cesty přímo v hlavním procesu lze také hodnotit kladně a je i týmem vhodně použita, stejně tak některé poznámky pomáhají snadnější orientaci a pochopení případů, nicméně v některých případech způsobují duplicitu informací. Odchylky v názvosloví jsou diskutabilní, označování scénáře procesem je otázkou vkusu a použití, ovšem nahrazení názvu „Rozšíření“ alternativním nebo chybovým procesem přispívá k přehlednosti.

Tato práce doporučuje řadu úprav v analyzovaných případech, které by jistě zvýšily přehlednost a opravily řadu chyb, kterých se autoři dopustili.

### 4.3.7 Příklad upraveného případu užití CoCoMe

Jako příklad případu užití upraveného podle daných doporučení byl zvolen případ užití číslo 4 - Přijetí objednaného zboží, který zde byl umístěn celý a byl sjednocen s „rozšířením“ případu užití číslo 8.

***Brief Description:*** Ordered products which arrive at the Store have to be checker for correctness and inventoried.

***Primary Actor:*** Stock Manager

***Goal in context:*** Stock Manager needs to keep the Trading System Inventory updated after ordered product arrived.

***Precondition:*** The Store Client was started and the part Inventory of the Trading System is available.

***Trigger:*** Ordered products arrive at the stock attached by an order identifier which has been assigned during the ordering process. [n4-1]

***Postcondition:*** The Inventory is updated with the ordered products.

***Standard scenario***

1. The Stock Manager checks the delivery for completeness and correctness. [p4-1, t42-1] In the case of correctness, the Stock Manager enters the order identifier and presses the button Roll in received order. [t43-1]

2. The Trading System updates the Inventory [t44-1] and notifies Stock Manager that the Inventory was updated.

***Alternative or Exceptional scenario***

– In step 1: Delivery not complete or not correct. [p4-2]

1. Stock Manager sends the products back to the supplier and has to wait until a correct and complete delivery has arrived at the stock.
2. New start of use case.

– In step 1: Moved products (according to UC8) arrive at the Store.

1. Stock Manager doesn't have to check the delivery but only count the incoming amount and enter it to the Trading System.
2. If the entered amount of an incoming product is larger than the amount accounted in the Inventory, System rejects the input.
3. Stock Manager has to re-enter the incoming amount.

Položka zúčastnění aktéři byla přejmenována na primární aktér (Primary actor) a byl přidán kontext cíle (Goal in context). Původní první bod procesu nahradil spouštěč a body kontroly a zadání přijetí objednávky byly sjednoceny do jednoho, aby vznikla několikrát zmiňovaná vazba aktér-systém.

Standardní proces dostal název Standardní scénář (Standard scenario) a stejně tak bylo slovo proces nahrazeno v názvu rozšíření. To bylo rozšířeno o případ, kdy dorazí zboží z jiné prodejny podle případu užití 8.

## 4.4 Shrnutí

Kapitola Vlastní práce ukázala, že vytvoření dokumentace pro již existující projekty má svá úskalí, která ztěžují začátek prací na dokumentaci. Trochu paradoxní výhodou pak je, že u existujícího řešení lze snáze narazit na chybu systému, a tím pádem ji i zachytit.

Případy užití by měly působit dojmem jednoduchého popisu s použití jednoduchého srozumitelného jazyka, ovšem jejich tvorba jednoduchá není Právě proto, aby bylo možné dosáhnout takovéto jednoduchosti, je třeba být velice precizní a dívat se na vlastní případy užití cizíma nezasvěcenýma očima.

Analýza případů užití CoCoME ukázala, že je možné dělat případy užití různým způsobem a zvolit různé šablony. Při porovnání šablony s šablonou A. Cockburna se však ukázalo, že ne všechny šablony jsou ideální a vhodné. Celkově je třeba hodnotit právě Cockburnovu šablonu za vhodnější. Analýza jednotlivých případů užití navíc objevila, že i dobré nápady je třeba správně konkrétně aplikovat, viz položka stručný popis, a že ani tvorba scénářů není jednoduchou záležitostí. Na konci analýzy pak byla názorně aplikována všechna doporučení na jeden z případů užití a náležitě okomentována.

# 5. Závěr

Cílem práce bylo seznámit se s hlavními způsoby specifikace softwarových požadavků na softwarové systémy, které jsou popsány v druhé kapitole práce s názvem „Specifikace požadavků softwaru“. Jsou zde typy požadavků, jejich sběr a detailněji i problematika samotných případů užití a pravidla použití jejich jednotlivých položek podle šablony A. Cockburna. Dalším úkolem bylo prozkoumat vybrané nástroje pro modelování a specifikaci požadavků, což v práci následuje s ukázkami z programů.

Kapitola „Vlastní práce“ pak ukazuje praktické použití případů užití a různá jeho úskalí. V této kapitole nalezneme postup tvorby specifikace pro systém SPOT a analýzu případů užití jiných autorů, ze které vyplynula řada návrhů na jejich zlepšení, které byly aplikovány na jeden ukázkový případ užití.

Přílohou práce jsou vypracované samotné případy užití a UML diagram. Je možné zde najít i celé analyzované případy užití CoCoME a jejich UML.

V práci se podařilo názorně aplikovat získané znalosti o tvorbě případů užití a jejich analýze. Ukázalo se při mnoha konzultacích, že tvorba případů užití není jednoduchá, je třeba být precizní a všímavý a téměř nic v systému nepovažovat za samozřejmost, což pro mě bylo velkým poučením. Nicméně výsledkem vznikly přehledné a snad také snadno pochopitelné případy užití a doporučení pro jejich tvorbu a jejich šablony.

Celkově formu dokumentace případy užití, kdy se nezachycuje chování systému, ale jeho průchod k dosažení aktérových cílů, považuji za velmi dobrou. Je škoda, že je v současné době válcována agilními metodikami, které obvykle takto podrobné dokumentaci neholdují.

# Přehled použitých zkratek

CoCoME – The Common Component Modeling Example neboli příklad modelování komponent. Jedná se o ukázku aplikace, která poskytuje případy užití k výzkumu softwarového modelování.

DPH – Daň z přidané hodnoty

DSP – Dokument specifikace požadavků

EA – Enterprise Architect (program)

FURPS+ – Metoda požívána pro hodnocení kvality softwarových produktů. Zkratka vznikla z prvních písmem slov: Funkčnost (Functionality), Použitelnost (Usability), Spolehlivost (Reliability), Výkon (Performance) a Udržovatelnost (Supportability) a pod znakem + se skrývají další požadavky.

IEEE – Institut pro elektrotechnické a elektronické inženýrství

IT – Informační technologie

MD – MagicDraw(program)

SPOT – Slovník překladů odborné terminologie

UC – Use case, v překladu Případ užití

UML – Unified Modeling Language (Jednotný modelovací jazyk)

WiFi – „Wireless fidelity“ technologie bezdrátového spojení v počítačových sítích

# Citovaná literatura

**Armour, Frank a Miller, Granville. 2001.** *Advanced Use Case Modeling: Software Systems.* Boston : Addison-Wesley, 2001.

**Booch, Grady, Rumbaugh, James a Jacobson, Ivar. 1999.** *The Unified Modeling Language User Guide.* Boston : Addison-Wesley, 1999.

**Bulej, Lubomir, a další.** CoCoME. *SourceForge.net.* [Online] [Citace: 15. Červen 2014.] http://sourceforge.net/projects/cocome/.

**Cockburn, Alistair. 2005.** *Use Cases: jak efektivně modelovat aplikace.* Brno : Computer Press, 2005.

**—. 2001.** *Writing effective use cases.* Boston : Addison-Wesley, 2001.

**IEEE. 1998.** *IEEE Recommended Practice for Software Requirements Specifications.* [Dokument] New York : autor neznámý, 1998. 0-7381-0332-2.

**—. 1990.** *IEEE Standard Glossary of Software Engineering Terminology.* New York : autor neznámý, 1990. 978-0-7381-0391-4.

**Lawrence, Brian. 1997.** Requirements Happens... *American Programmer.* 4, 1997, 10.

**No Magic Inc. 2014.** *MagicDraw - User manual.* [Dokument] 2014.

**Object Management Group. 2013.** OMG Unified Modeling Language 2.5. *Object Management Group.* [Online] 5. září 2013. http://www.omg.org/spec/UML/2.5/Beta2/PDF.

**Rausch, Andreas. 2008.** Strana 16–53. *Common Component Modeling Example.* Berlin Heidelberg : Springer-Verlag, 2008.

**Wiegers, Karl E. 2003.** *Software Requirements, Second Edition.* Redmond : Microsoft Press, 2003.

# Seznam obrázků

# Přílohy

## A. IEEE Std 830-1998 Šablony specifických požadavků dokumentu specifikace požadavků [IEEE 1990]

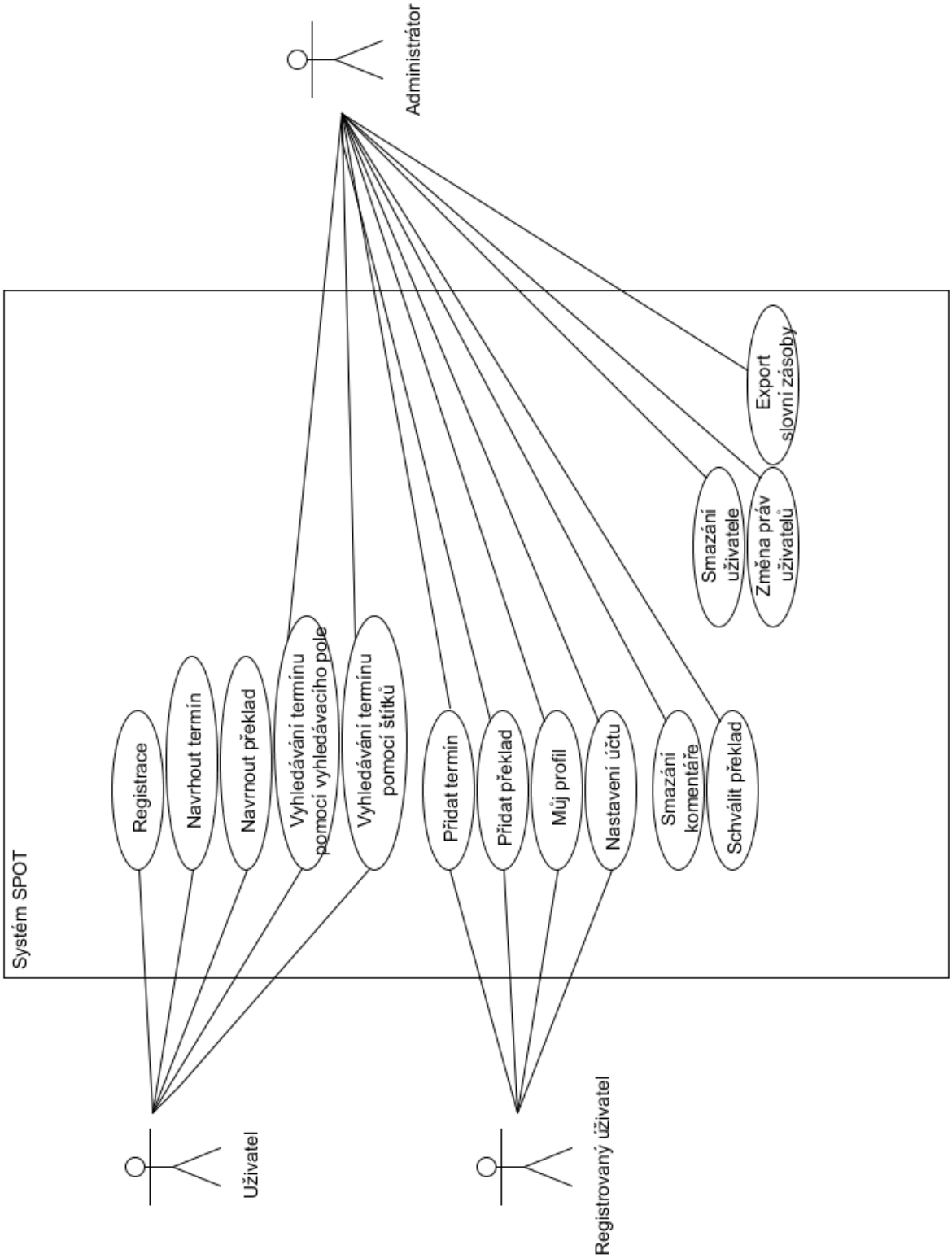
### Alternativní rozdělení podle módů

3. Specifické požadavky
    3.1 Požadavky na vnější prostředí
        3.1.1 Uživatelské prostředí
        3.1.2 Hardwarové rozhraní
        3.1.3 Softwarové rozhraní
        3.1.4 Komunikační rozhraní
    3.2 Funkční požadavky
        3.2.1 Mód 1
            3.2.1.1 Funkční požadavek 1.1
        .
        .
        .
            3.2.1.*n* Funkční požadavek 1.*n*
        3.2.2 Mód 2
        .
        .
        .
        3.2.*m* Mód *m*
            3.2.*m.*1 Funkční požadavek *m.*1
            .
            .
            .
            3.2.*m.n* Funkční požadavek *m.n*
    3.3 Výkonnostní požadavky
    3.4 Omezující podmínky
    3.5 Vlastnosti systému
    3.6 Další požadavky

### Rozdělení specifických požadavků podle uživatelských tříd

3. Specifické požadavky
    3.1 Požadavky na vnější prostředí
        3.1.1 Uživatelské rozhraní
        3.1.2 Hardwarové rozhraní
        3.1.3 Softwarové rozhraní
        3.1.4 Komunikační rozhraní
    3.2 Funkční požadavky
        3.2.1 Uživatelská třída 1
            3.2.1.1 Funkční požadavek 1.1
            .
            .
            .
            3.2.1.*n* Funkční požadavek 1.*n*
        3.2.2 Uživatelská třída 2
        .
        .
        .
        3.2.*m* Uživatelská třída *m*
            3.2.*m.*1 Funkční požadavek *m.*1
            .
            .
            .
            3.2.*m.n* Funkční požadavek *m.n*
        3.3 Výkonnostní požadavky
        3.4 Omezující podmínky
        3.5 Vlastnosti systému
        3.6 Další požadavky

## B. UML diagram systému spot.zcu.cz



# C. Vypracované ukázkové případy užití

## Searching term using a search field / UC 001

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: User

Goal in context: User searches term and its translation using system or he could require adding term or translation.

Precondition: none

Trigger: User wants to search term in system database.

Postcondition: None

Main success scenario:

1. User enters searching term (at least 2 characters) into the system search field.
2. The System shows the first page of all found terms and their translations and possibly found tags.
3. User chooses on some of pages searching term.
4. The System shows the details of the searched term
   (Abbreviation, Added by, Status, Usage and Description) and the list of translations.
5. User chooses from the offered translations the most appropriate one.
   User could unwrap details of translation, whereas Evaluation of translation, Status, name of a user how added the translation, Source, Tags and Description.

Extensions:

2a. Entered term isn't found in the system.

2a1 The System informs the user**:** Form for adding the term into database is below.

2a2 User could add the term / UC 005 or if he is not logged in suggest the term / UC 006.

4a. There is no translation of the term in the system

4a1 User could add the translation / UC 003 in case he knows it or if he is not logged in could suggest the translation / UC 004.

### Searching term using tags / UC 002

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: User

Goal in context: User searches a term and its translation using system or he is able to require adding term or translation.

Precondition: User is on the page "Tags" in the SPOT system.

Trigger: User wants to search term by using tags in system database.

Postcondition: None

Main success scenario:

1. User chooses tag where is according to him searching term.
2. The system shows first page of the list of tagged terms and their translations.
3. User chooses on some of pages a term.
4. The system shows the page with translations of searching term and the term details to the user
   (Abbreviation, Added by, Status, Usage and Description).
5. User chooses the most appropriate one from all offered translations.
   User could unwrap details of translations whereas Evaluation of translation, Status, name of a user how added the translation, Source, Tags and Description.

Extensions:

1a In the system isn't appropriate tag.

1a1 User could searches term using a search field / UC 001
1a2 User could add tag to searching term.

2a. In chosen tag isn't searching term.

2a1 User could searches term using a search field / UC 001
2a2 User could add tag to searching term if he is logged in.

3a. There is no required translation in the system

3a1 User could if he knows add the translation / UC 003 or if he is not logged in could suggest the translation / UC 004.

### Adding a translation / UC 003

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: Registered User

Goal in context: User adds a translation in the system database to make it possible to search.

Precondition: User is logged in the system and knows translation of the term.

Trigger: User searches term using search field / UC 001 or searches term using tags/ UC 002 and didn't found the most appropriate translation of the term.

Postcondition: Added translation is saved in database.

Main success scenario:

1. User adds translation by clicking to Add a translation button and by filling the form (Czech translation, Type of translation and Description).
2. System announces adding translation to database.

Extensions:

1a. The term already have appropriate translation.

1a1 System announces that adding translation is already in database.
1a2 End of this use case.

### Suggesting a translation / UC 004

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: User

Goal in context: User suggests a translation to the system database.

Precondition: User knows translation of the term

Trigger: User searches term using search field / UC 001 or searches term using tags / UC 002 and didn't found the most appropriate translation of the term.

Postcondition: Added suggestion is saved in database.

Main success scenario:

1. User adds translation by clicking to add a translation button and by filling the form (Translation suggestion and E-mail).
2. System announces adding suggestion of translation to database.

Extensions:

1a. Term already have appropriate translation.

1a1 System announces that suggesting translation is already in database.
1a2 End of this use case.

### Adding a term / UC 005

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: Registered User

Goal in context: User adds a term in the system database to make it possible to search.

Precondition: User is logged in the system.

Trigger: User searches term using search field / UC 001 and didn't found the term.

Postcondition: Added term is saved in database.

Main success scenario:

1. User fills the form
   (Chooses and fills word or abbreviation and if knows Translation suggestion).
2. System announces adding term to database.

### Suggesting a term / UC 006

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: User

Goal in context: User suggests a term to the system database.

Precondition: None

Trigger: User searches term using search field / UC 001 didn't found the term.

Postcondition: Added suggestion is saved in database.

Main success scenario:

1. User fills the form
   (Chooses and fills word or abbreviation, if knows Translation suggestion and Email).

2. System announces adding term to database.

### Registration / UC 007

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: User

Goal in context: User signs up into the system database to have access to more system functions.

Precondition: User is on the registration page.

Trigger: User wants to sign up into the spot system.

Postcondition: User has been added to the database and the account has been activated.

Main success scenario:

1. User fills in the registration form
   (Login, Password, again password, E-mail and determines if the e-mail is public or not and enters the verification text.
2. System notifies user that the form was send and prompts to activation per e-mail.
3. User activates account using link in the received e-mail.
4. System notifies that the account is activated.

Extensions:

1a. User login is already in the system.

1a1 System notifies it to the user.

1a2 User enters other login.

1b Passwords do not match

1b1 System notifies that.

1b2 User corrects an error.

1c The verification text is incorrectly filled.

1c1 System notifies it and generates a new one.

1c2 User rewrites a new verification text.

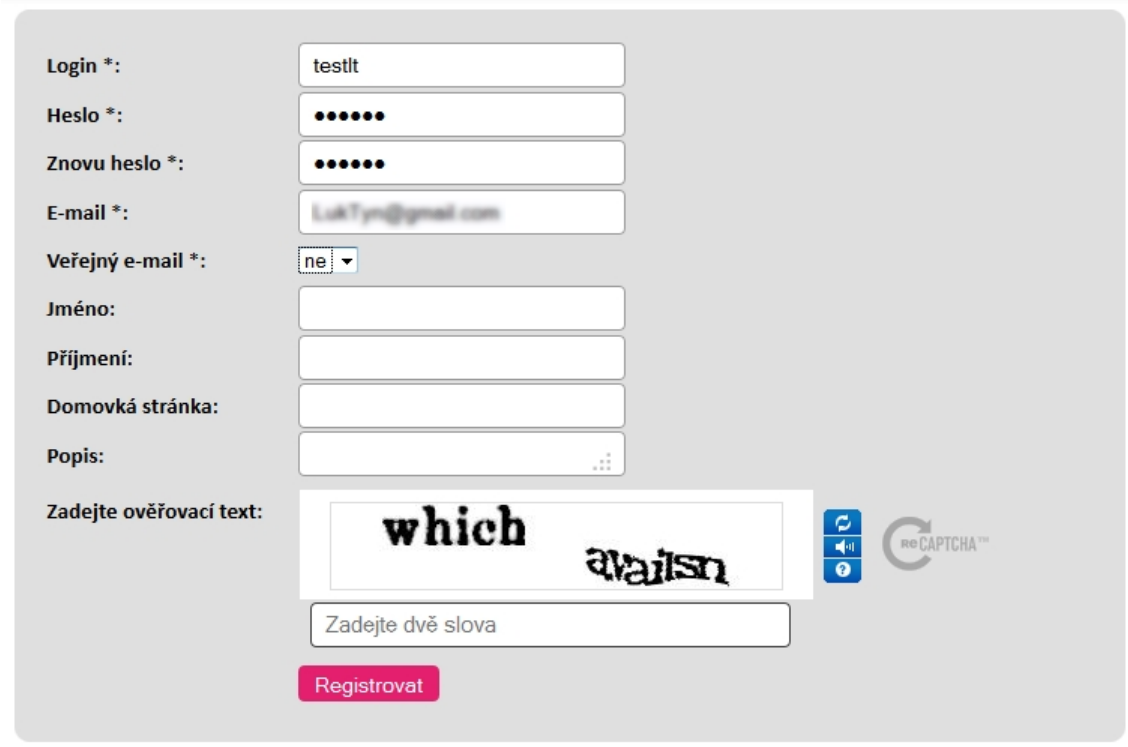
2a System couldn’t send the activation mail

2a1 System notifies it and prompts user to register later.

3a User hasn't activated an account.

3a System informs the user that his account isn’t activated while he is trying to log in.

Annex (Příloha):




Podoba registračního formuláře

### Account setting / UC 008

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: Registered User

Goal in context: User changes his details in system database to keep them actual

Precondition: User is logged in and on User account setup page.

Trigger: User decides to change his details in his user account.

Postcondition: All new details had been saved in the system database.

Main success scenario:

1. User fills form items he wants to change. He has to enter Old / Actual password. User could change Password if it is necessary to enter two-times, Email and chooses if it has to be public or not, First name and Last Name, Homepage and Description.
2. System notifies to the user that form was sent.

Extensions:

1a Old / Actual password is incorrect

1a1 System notifies that.

1b2 User corrects an error.

1b New passwords do not match

1b1 System notifies that.

1b2 User corrects an error.

### My profile/ UC 009

Scope: *The dictionary system of technical terminology (spot.zcu.cz)*

Level: User goal

Primary actor: Registered User

Goal in context: User could see all of his post in system.

Precondition: User is logged in.

Trigger: User wants to see his profile.

Postcondition: None

Main success scenario:

1. User goes to My profile page.
2. System shows to user added words, abbreviation and translation and their status, comments and projects. System shows user's account data and Top user list and user's rank and points count.

Extensions:

2a User could click on My added words, My added abbreviations and My added translations to see more terms.

2a1 System shows list off all corresponding terms, translations and their status.

2b User could click on My comments.

2b1 System shows list off all user's comments and terms to which they relate.

2c User could click on My projects

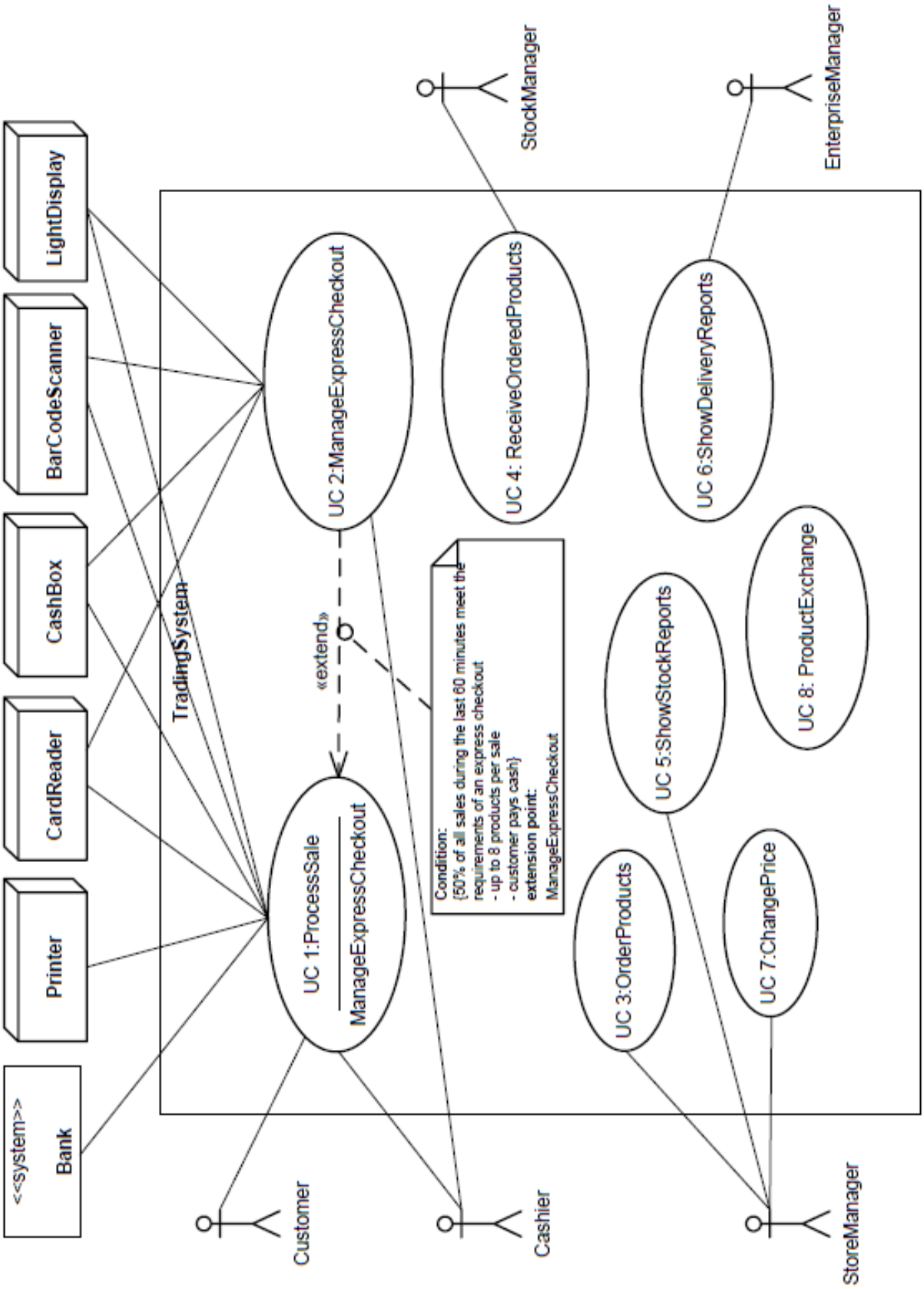
2c1 System shows list of all created projects and form for creation of project.

2c2 User could by filling the project name create new project.
2c3 System notifies project creation.

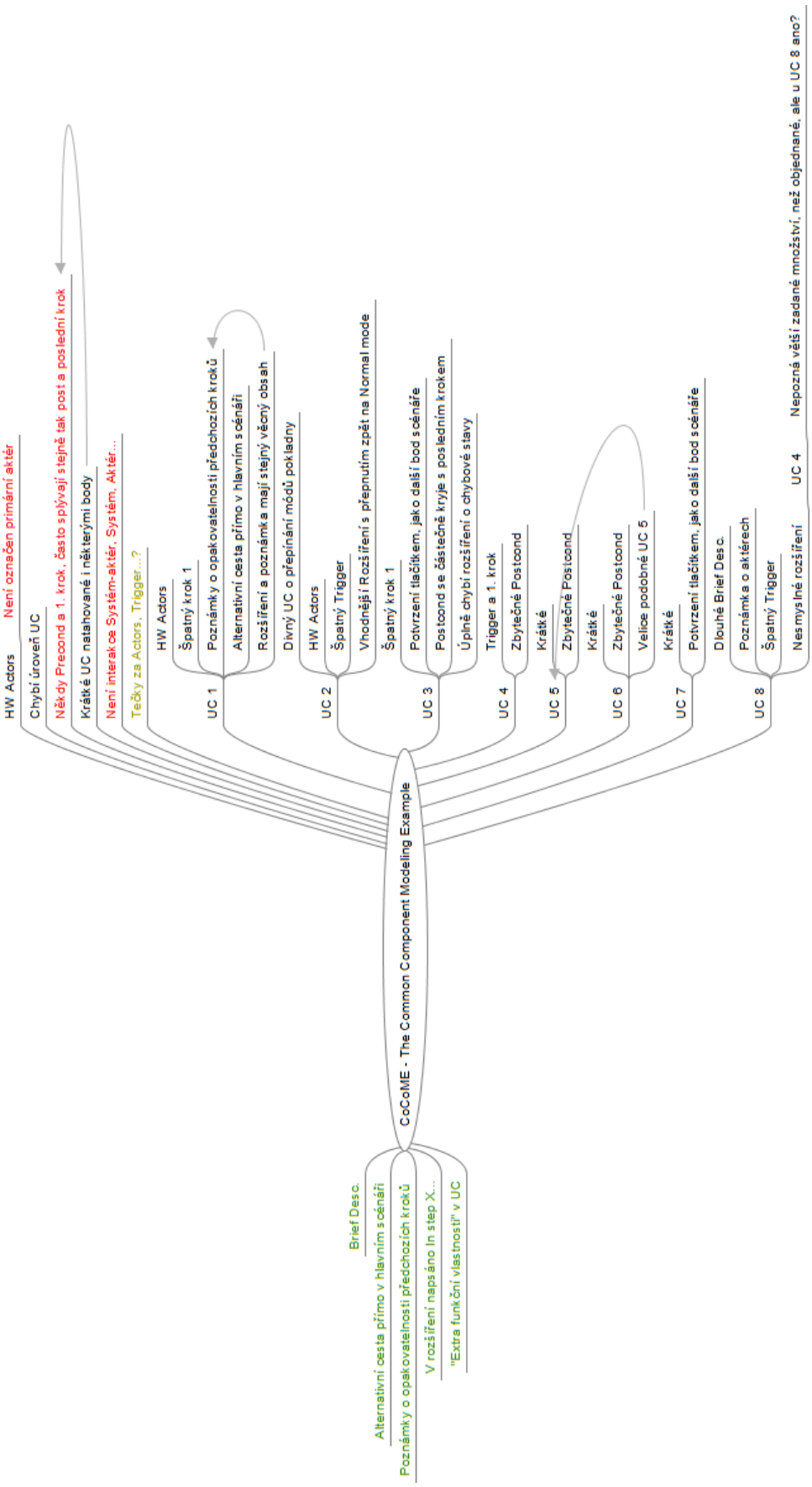
2d User finds outdated personal data or wants to change account details.

2d1 User could change Account settings / UC 008 by clicking on Edit my data.

## D. UML diagram CoCoME [Rausch 2008]



## E. Myšlenková mapa CoCoME



# F. Případy užití CoCoME [Rausch 2008]

## UC 1 - Process Sale

Brief Description. At the Cash Desk the products a Customer wants to buy are detected and the payment - either by credit card or cash - is performed.

Involved Actors. Customer, Cashier, Bank, Printer, Card Reader, Cash Box, Bar Code Scanner, Light Display

Precondition. The Cash Desk and the Cashier are ready to start a new sale.

Trigger. Coming to the Cash Desk a Costumer wants to pay his chosen product items.

Postcondition. The Customer has paid, has received the bill and the sale is registered in the Inventory.

Standard Process

1. The Customer arrives at the Cash Desk with goods to purchase. [arr1]

2. The Cashier starts a new sale by pressing the button Start New Sale at the Cash Box. [t12-1]

3. The Cashier enters the item identifier. This can be done manually by using the keyboard of the Cash Box [p13-1, t13-1] or by using the Bar Code Scanner [p13-2, t13-2].

4. Using the item identifier the System presents the corresponding product description, price, and running total. [t14-1]

The steps 3-4 are repeated until all items are registered. [n11-2]

5. Denoting the end of entering items the Cashier presses the button Sale Finished at the Cash Box. [t15-1]

(a) To initiate cash payment the Cashier presses the button Cash Payment at the Cash Box. [p15-1,t15a-1]

i. The Customer hands over the money for payment. [t15a1-1]

ii. The Cashier enters the received cash using the Cash Box and confirms this by pressing Enter. [t15a2-1]

iii. The Cash Box opens. [t15a3-1]

iv. The received money and the change amount are displayed [t15a4-1], and the Cashier hands over the change. [t15a4-2]

v. The Cashier closes the Cash Box. [t15a5-1]

(b) In order to initiate card payment the Cashier presses the button Card Payment at the Cash Box. [p15-2, t15b-1]

i. The Cashier receives the credit card from the Customer [t15b1-1] and pulls it through the Card Reader. [t15b1-2]

ii. The Customer enters his PIN using the keyboard of the card reader and waits for validation. [t15b2-1]

The step 5.b.ii is repeated until a successful validation or the Cashier presses the button for cash payment. [t15b2-2, n15b2-1]

6. Completed sales are logged by the Trading System and sale information are sent to the Inventory in order to update the stock. [t16-1]

7. The Printer writes the receipt and the Cashier hands it out to the Costumer. [t17-1]

8. The Customer leaves the Cash Desk with receipt and goods.

Alternative or Exceptional Processes

– In step 3: Invalid item identifier if the system cannot find it in the Inventory. [p13-4]

1. The System signals error and rejects this entry. [t13-3]

2. The Cashier can respond to the error as follows:

(a) It exists a human-readable item identifier: [p13-5]

i. The Cashier manually enters the item identifier. [t13-4]

ii. The System displays the description and price. [t14-1]

(b) Otherwise the product item is rejected. [p13-6]

– In step 5.b: Card validation fails. [p15b2-2]

1. The Cashier and the Customer try again and again.

2. Otherwise the Cashier requires the Customer to pay cash.

– In step 6: Inventory not available. [p16-1]

The System caches each sale and writes them into the Inventory as soon as it is available again. [t161-1]

## UC 2 - Manage Express Checkout

Brief Description. If some conditions are fulfilled a Cash Desk automatically switches into an express mode. The Cashier is able to switch back into normal mode by pressing a button at his Cash Desk. To indicate the mode the Light Display shows different colors.

Involved Actors. Cashier, Cash Box, Light Display, Card Reader

Precondition. The Cash Desk is either in normal mode and the latest sale was finished (case 1) or the Cash Desk is in express mode (case 2).

Trigger. This use case is triggered by the system itself.

Postcondition. The Cash Desk has been switched into express mode or normal mode. The Light Display has changed its color accordingly.

Standard Process

1. The considered Cash Desk is in normal mode [p2-1] and just finished a sale which matches the condition of an express checkout sale. Now 50% of all sales during the last 60 minutes fulfill the condition for an express checkout.

(a) This Cash Desk, which has caused the achievement of the condition, is switched into express mode. [t21a-1]

(b) Furthermore the corresponding Light Display is switched from black into green to indicate the Cash Desk's express mode. [t21b-1]

(c) Paying by credit card is not possible anymore. [t21c-1]

(d) The maximum of items per sale is reduced to 8 and only paying by cash is allowed. [t21d-1]

2. The Cash Desk is in express mode [p2-2] and the Cashier decides to chase back into normal mode.

(a) The Cashier presses the button Disable Express Mode. [t22a-1]

(b) The color of the Light Display is changed from green into black color. [t22b-1]

(c) Cash and also card payment is allowed and the Costumer is allowed to buy as much goods as he likes. [t22c-1]

### UC 3 - Order Products

Brief Description. The Trading System provide the opportunity to order product items.

Involved Actors. Store Manager

Precondition. An Overview over the Inventory is available and the Store Client was started.

Trigger. The Store Manager decided to buy new product items for his store.

Postcondition. The order was placed and a generated order identifier was presented to the Store Manager.

Standard Process

1. A list with all products [n3-1] and a list with products running out of stock are shown. [n3-2, p3-1, t31-1]

2. The Store Manager chooses the product items to order and enters the corresponding amount. [t32-1]

3. The Store Manager presses the button Order at the Store Client's GUI. [t33-1]

4. The appropriate suppliers are chosen and orders for each supplier are placed. An order identifier is generated for each order and is shown to the Store Manager. [t34-1, t34-2, t34-3]

### UC 4 - Receive Ordered Products

Brief Description. Ordered products which arrive at the Store have to be checker for correctness and inventoried.

Involved Actors. Stock Manager

Precondition. The Store Client was started and the part Inventory of the Trading System is available.

Trigger. The ordered products arrive at the Store.

Postcondition. The Inventory is updated with the ordered products.

Standard Process

1. Ordered products arrive at the stock attached by an order identifier which has been assigned during the ordering process. [n4-1]

2. The Stock Manager checks the delivery for completeness and correctness. [p4-1, t42-1]

3. In the case of correctness, the Stock Manager enters the order identifier and presses the button Roll in received order. [t43-1]

4. The Trading System updates the Inventory. [t44-1]

Alternative or Exceptional Processes

– In step 2: Delivery not complete or not correct. [p4-2]

The products are sent back to the supplier and the Stock Manager has to wait until a correct and complete delivery has arrived. This action does not recognized by the System.

### UC 5 - Show Stock Reports

Brief Description. The opportunity to generate stock-related reports is provided by the Trading System.

Involved Actors. Store Manager

Precondition. The reporting GUI at the Store Client has been started.

Trigger. The Store Manager wants to see statistics about his store.

Postcondition. The report for the Store has been generated and is displayed on the reporting GUI.

Standard Process

1. The Store Manager enters the store identifier and presses the button Create Report. [t51-1]

2. A report including all available stock items in the store is displayed. [t52-1]

### UC 6 - Show Delivery Reports

Brief Description. The Trading System provides the opportunity to calculate the mean times a delivery from each supplier to an considered enterprise takes.

Involved Actors. Enterprise Manager

Precondition. The reporting GUI at the Store Client has been started.

Trigger. The Enterprise Manager wants to see statistics about the enterprise.

Postcondition. The report for the Enterprise has been generated and is displayed to the Enterprise Manager.

Standard Process

1. The Enterprise Manager enters the enterprise identifier and presses the button Create Report. [t61-1]

2. A report which informs about the mean times is generated. [t62-1]

### UC 7 - Change Price

Brief Description. The System provides the opportunity to change the sales price for a product.

Involved Actors. Store Manager

Precondition. The store GUI at the Store Client has been started.

Trigger. The Store Manager wants to change the sales price of a product for his store.

Postcondition. The price for the considered product has been changed and it will be sold with the new price now.

Standard Process

1. The System presents an overview over all available products in the store. [t71-1]

2. The Store Manager selects a product item [t72-1] and changes its sales price. [t72-2]

3. The Store Manager commits the change by pressing ENTER. [t73-1]

### UC 8 - Product Exchange (on low stock) Among Stores

Brief Description. If a store runs out of a certain product (or a set of products; “required good”), it is possible to start a query to check whether those products are available at other Stores of the Enterprise (“providing Stores”). Therefore the Enterprise Server and the Store Servers need to synchronize their data on demand (one scheduled update per day or per hour is not sufficient). After a successful query the critical product can be shipped from one to other Stores. But it has to be decided (using heuristics to compute the future selling frequency), whether the transportation is meaningful. For example, if the product is propably sold out at all Stores within the same day, a transportation does not make sense.

Expressed in a more technical way one Store Server is able to start a query at the Enterprise Server. The Enterprise Server in turn starts a query for products available at other Stores. As the Enterprise Server does not have the current global data for Stores at any time (due to a write caching latency at the Store Servers) the Enterprise Server has to trigger all Store Servers to push their local data to the Enterprise Server.

Involved Actors. This use case is not an end-user use case. Only servers are involved.

Precondition. The Store Server with the shortage product is able to connect to the Enterprise Server.

Trigger. This use case is triggered by the system itself.

Postcondition. The products to deliver are marked as incoming or unavailable, respectively, in the according Stores.

Standard Process

1. A certain product of the Store runs out.

2. The Store Server recognizes low stock of the product. [t82-1]

3. The Store Server sends a request to the Enterprise Server (including an identification of the shortage products, and a Store id) [t83-1]

4. The Enterprise Server triggers all Stores that are “near by” (e. g. ¡300 km) the requiring store, to flush their local write caches. So the Enterprise Server database gets updated by the Store Server. [t84-1, t84-1]

5. The Enterprise Server does a database look-up for the required products to get a list of products (including amounts) that are available at providing Stores. [t85-1]

6. The Enterprise Server applies the “optimization criterion” (specified above) to decide, whether it is meaningful to transport the shortage product from one store to another (heuristics might be applied to minimize the total cosi of transportation). This results in a list of products (including amounts) per providing store that have to be delivered to the requiring Store. [t86-1]

7. The Store Server, initially sending the recognition of the shortage product, is provided with the decision of the Enterprise Server. [t87-1]

(a) The required product is marked as incoming. [t87-2]

8. The Store Server of a near by Store is provided with information that it has to deliver the product. [t88-1]

(a) The required product is marked as unavailable in the Store. [t88-2]

Alternative or Exceptional Processes

– The Enterprise Server is not available: The request is queued until the Enterprise Server is available and then is send again.

– One or more Store Servers are not available: The Enterprise Server queues the requests for the Store Servers until they are available and then resend them.

If a Store Server is not available for more than 15 minutes the request for this Server is canceled. It is assumed, that finally unavailable Store Servers do not have the required product.

### Extension on use case 8 - Remove Incoming Status

Brief Description. If the first part of use case 8 (as described above) has passed, for moved products an amount marked as incoming remains at the Inventory of the Store receiving the products. An extension allows to change that incoming mark via a user interface at the Store Client if the moved products arrive at a Store.

Precondition. The Inventory is available and the Store Client has been started.

Trigger. The moved products (according to UC8) arrive at the Store.

Postcondition. For the amount of incoming products the status ”incoming” is removed in the Inventory.

Standard Process

1. The products arrive at the stock of the Store.

2. For all arriving products the Stock Manager counts the incoming amount.

3. For every arriving product the Stock Manager enters the identifier and its amount into the Store Client.

4. The system updates the Inventory.

Alternative or Exceptional Processes

– If the entered amount of an incoming product is larger than the amount accounted in the Inventory, the input is rejected. The incoming amount has to be re-entered.