Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Generování reportů z BORM procesních diagramů

Jan Valášek

Vedoucí práce: Mgr. Martin Podloucký

12. května 2015

Poděkování

Děkuji vedoucímu práce, Mgr. Martinu Podlouckému, za poskytnuté rady a konzultace při realizaci této práce. Dále děkuji spolužákům Peteru Uhnákovi a Janu Blizničenkovi za rychlou pomoc s řešením všech technických problémů během vývoje programu a Radku Doubravovi za finální proofreading. Asi největší dík patří Míše Němcové, za její trpělivost a veškerou podporu a také rodině za to, že mi umožnila vůbec studovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2015 .....

Odkaz na tuto práci

Valášek, Jan. *Generování reportů z BORM procesních diagramů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato práce se zaměřuje na návrh, analýzu a implementaci exportovacího nástroje, který provádí konverzi BORM procesních diagramů do textové podoby. Nástroj je implementován v jazyce Smalltalk v prostředí Pharo, jako součást editoru diagramů DynaCASE.

Při vývoji tohoto nástroje byl kladen důraz především na možnost snadného provádění exportu uživatelem, vytvoření rozhraní pro komunikaci s programem DynaCASE, správnou programovou interpretaci diagramu a jeho export do přehledné formy.

Výsledky exportu jednoduchého diagramu do jednotlivých formátů se nacházejí v příloze na konci této práce.

Klíčová slova Procesní diagram, BORM, export diagramu, textový report

Abstract

The thesis deals with designing, analysing and implementing a tool for exporting BORM porcess diagrams into text. The tool is implemented in the Smalltalk programming language in the Pharo IDE as a part of DynaCASE diagram editor.

The developement was focused on easy usage of this tool, creating an interface for communication with DynaCASE, correct interpretation of given diagrams and their export to clearly readable text.

The results of exporting a simple diagram to text form are in the appendix of the thesis.

Keywords Process diagram, BORM, export of diagram, text report

Obsah

Seznam obrázků

Úvod

Analýza a definování procesů jsou důležité součásti procesního řízení[1], jehož význam pro firmy a organizace stále roste [3]. Tento vzestup v dnešní době umožňují mimo jiné informační technologie, které jsou využívány v různých procesních disciplínách a firemním managementu. Například Business Process Management je rozšířený koncept orientovaný na management firem, který zahrnuje mimo jiné i modelování a simulaci business procesů [4].

Metody pro popis procesů

Pro popis procesů jsou využívány různé metody a notace **BPM** (Business process modeling) [5], například:

- **BPMN** (Business Process Model and Notation) [6]
- **BORM** (Business Object Relation Model) [1]
- **DEMO** (Design & Engineering Methodology for Organizations) [7]

V rámci bakalářské práce využívám jako zdroj dat pro tvorbu diagramů právě metodu BORM. Pomocí této metody lze vytvářet procesní diagramy, popisující konkrétní business procesy.

Další významná metoda v této oblasti je metoda DEMO, ale narozdíl od BORMu nepodporuje například simulaci procesů [4].

Tvorba diagramů

Pomocí těchto metodik lze jednak analyzovat stávající procesy, ale i určovat a navrhovat procesy nové [5]. Nejčastěji ve formě diagramů a grafů. K tomuto

[1] soubor činností, které se týkají plánování a sledování výkonnosti zejména realizačních firemních procesů

účelu slouží různé editační a vizualizační programy. V této bakalářské práci využívám nástroj DynaCASE.

DynaCASE je nástroj na tvorbu a správu různých druhů diagramů, například UML, ontoUML, konečných automatů a BORM procesních diagramů. Jeho implementace v jazyce Smalltalk funguje na platformě Pharo. Vznikl v rámci předmětu Softwarový projekt (BI-SP1) na Fakultě informačních technologií ČVUT, a je nadále vyvíjen v rámci skupiny CCMi[2].

Potřeba generování reportů

Aby mohl být nový nebo upravený proces uveden do praxe, je potřeba, aby každý účastník procesu znal doporučené postupy a omezení, která musí dodržovat. Samotný diagram lze poměrně snadno v editoru vytvořit či spravovat. Například pro manažera je navíc takový diagram velmi přehledný, protože v něm vidí všechny participanty s jejich vzájemnými vztahy.

Když ovšem potřebuje ten který participant znát jen postup své práce, tak je vhodnější a praktičtější vybrat z diagramu pouze informace týkající se přímo daného participanta a ty mu předat místo kompletního diagramu, který může obsahovat mnoho redundantní informací. Dále je důležité, aby účastník procesu přesně věděl jak správně postupovat, kdy je potřeba komunikovat s ostatními participanty a co komu v dané fázi procesu předat.

K tomuto účelu slouží jako vhodná pomůcka **textové reporty** vygenerované pro konkrétní účastníky procesu. Textové reporty obsahují přehledné informace a pokyny, kterými se může participant řídit.

Postup při tvorbě reportu

Procesní diagramy zpracovávané v rámci této bakalářské práce jsou vytvořeny v nástroji DynaCASE, kde je implemetována metoda BORM jako jeden z jeho balíčků tříd. Generátor reportů operuje nad tímto balíčkem a dokáže z něj těžit textová i logická data, která převádí do mezijazyka Pillar [3]. Z Pillaru jsou pak dále produkovány výsledné soubory reportu v požadovaném formátu.

Členění práce

V úvodu této práce byl krátce popsán význam business procesů a různých metod pro jejich vyjádření. Dále následovalo vysvětlení potřeby generování reportů z diagramu a nastínění postupu, jakým to lze provést.

Následující kapitola 1 zahrnuje stanovení cílů a popis problémů, které je potřeba v rámci bakalářské práce analyzovat a vyřešit.

[2] Centre for Conceptual Modelling and Implementation (`http://ccmi.fit.cvut.cz`)

[3] `http://smalltalkhub.com/#!/ Pier/Pillar`

Kapitola 2 je věnována přiblížení metody BORM. Stručně popisuje historii, jednotlivé součásti, a příklady využití této metody.

Navazující kapitola 3 se zabývá analýzou a návrhem možných řešení jak generovat textové reporty. Dále také výběrem vhodných nástrojů, popisem jednotlivých částí procesu exportu a vysvětlením potřeby obecného mezijazyka.

Důležité prvky realizace jsou obsaženy v kapitole 4, kde se nachází seznam použitých technologií, popis struktury programu a následné implementace.

Tuto práci uzavírá kapitola 5, zabývající se testováním výsledného programu. Nejprve z pohledu statické analýzy kódu a poté se zaměřuje na funkční a výkonostní testy.

Kapitola 1

Cíle a řešené problémy

Cílem práce je navrhnout a implementovat generátor textových reportů z procesních diagramů vytvořených metodou BORM. Provést analýzu daného problému, vybrat vhodné nástroje pro převod procesních modelů do textové podoby a nakonec navrhnout či vybrat vhodný obecný jazyk pro popis diagramů před samotným exportem do výsledného formátu. V implementační části pak naprogramovat převod procesního diagramu na dokument vyjádřený v tomto obecném jazyce a poté transformaci takového dokumentu do vybraných formátů, jako je HTML[4], LaTeX[5] či ODF[6]. Výsledkem práce bude plug-in pro některý z modelovacích nástrojů vyvíjených v rámci CCMi.

1.1 Postup generování textového reportu

Samotný postup generování reportu se skládá z několika základních částí, viz obrázek 3.2. Návrh a implementaci všech těchto částí do zásuvného modulu řeším v rámci této bakalářské práce.

1. Nejprve je potřeba programově zanalyzovat zadaný BORM diagram, všechny participanty, a vyhodnotit možné alternativy ve vykonávání procesu pro každého z participantů.

2. Následně musí program vytvořit obecnou datovou strukturu (*mezijazyka*) pro popis výsledného dokumentu, do které zadaný diagram převede.

3. Nakonec zásuvný modul převede dokument uložený v obecném mezijazyce do požadovaného výstupního formátu.

[4]HyperText Markup Language - Standartní jazyk pro tvorbu webových stránek
[5]LaTeX - Systém na přípravu a sazbu dokumentů
[6]OpenDocument Format - Otevřený formát dokumentu založený na XML formátu

1.2 Analýza diagramu

Jako první krok při generování reportu je potřeba správně načíst, zanalyzovat a interpretovat zadaný procesní diagram. Tento diagram je v systému reprezentován souborem různých typů objektů, vazeb mezi nimi a pravidly pro průchod diagramem. Diagram obsahuje jednotlivé účastníky (participanty) daného procesu. Každý participant má dále určené schéma stavů, aktivit (činností), komunikací s ostatními účastníky a různých podmíněných kroků. Součástí implementace je tedy algoritmus na vhodné procházení grafu při respektování vnitřních podmínek a jeho následný převod do textové podoby.

1.3 Výběr a použití vhodného mezijazyka

Aby nebylo potřeba programovat analýzu a zpracování dat z diagramu pro každý formát exportu zvlášť (viz obrázek 3.1), musíme navrhnout nebo nalézt vhodnou datovou strukturu pro uchování těchto dat v obecné podobě (viz obrázek 3.2). Tato struktura by měla být dostatečně obecný mezijazyk, aby stačilo jen definovat pravidla pro převod z obecného jazyka do požadované struktury výstupního formátu bez potřeby znalosti samotného datového modelu diagramu. Tedy bude stačit naprogramovat pouze jednou načtení diagramu a jeho interpretaci, nezávisle na počtu požadovaných formátů pro export. Tímto se také docílí snadné rozšiřitelnosti programu o další exportované formáty v budoucnu.

1.4 Export do požadovaného formátu

Reporty je vhodné vytvářet v různých formátech, podle konkrétních potřeb jednotlivých uživatelů, kteří aplikaci využívají a pro které jsou reporty určeny. Dále také může ovlivnit výběr formátu cílová platforma (systémy Microsoft Windows, Linux, Webové prostředí, fyzický výtisk na papíře...). Pro tisk se nejvíce hodí například formáty z programů LaTeX[7] či Microsoft Word[8]. Jednotlivé reporty se z něj dají převést do pdf nebo přímo vytisknout a používat i „offline“. K online distribuci jsou vhodné „webové jazyky“ jako například XML či HTML. Takové formáty se velmi hodí mimo jiné k distribuci prostřednictvím firemního intranetu nebo elektronickou poštou.

1.5 Požadavky na report

Report by měl sloužit jako osnova pro jednotlivé účastníky daných procesů. Výsledný report tedy bude obsahovat přehlednou posloupnost kroků, které je

[7] LaTeX – A document preparation system (`http://www.latex-project.org/`)
[8] MS Word (`https://products.office.com/en-us/word`)

potřeba provést konkrétním participantem procesu. Navíc co nejsrozumitelněji zpracuje různé možnosti větvení činností díky podmínkám v procesním diagramu, které musí nebo nemusí být splněny pro další pokračování. Poslední požadavek je snadná orientace ve výsledném reportu, což zahrnuje co nejméně složitých tabulek a minimální nutnost často přeskakovat v textu.

Kapitola 2

Procesní diagramy a metoda BORM

2.1 BORM

Business Objects Relation Modelling, zkráceně BORM, je komplexní metoda pro tvorbu konceptuálních modelů zaměřených na business procesy. Tato metoda je vhodná na propojení business a software engineeringu. [8]

2.1.1 Vznik

Vývoj této metody začal v roce 1993 jako součást grantového projektu VAPPIENS Britské rady (British Council). Vývoj je od roku 1996 podporován konzultační firmou Deloitte, kde se využívá v rozsáhlých poradenských projektech. Dnes je tato metoda dále vyvíjena členy CCMi[9]. CCMi se věnuje především analýze objektového chování - BORM Object Behaviour Analysis method - BOBA [9].

2.1.2 Popis metody [9]

Procesní diagram (viz obr. 2.1) obsahuje jednotlivé účastníky procesu neboli participanty. Každý participant má zadánu množinu aktivit a stavů, které jsou navzájem propojeny orientovanými hranami ve směru průběhu procesu. Aktivity mohou být navzájem provázány pomocí komunikace a datových toků mezi více participanty. Každý participant by měl mít právě jeden počáteční stav.

- **Diagram** - Obsahuje všechny potřebné informace o procesu a jeho participantech. Prostřednictvím diagramu lze přistupovat ke všem částem procesu a připravit dokument k exportu.

[9]Centre for Conceptual Modelling and Implementation (`http://ccmi.fit.cvut.cz`)

- **Participant** - Reprezentace osoby, organizace nebo systému účastnící se procesu (znázorněný šedým obdélníkem na obrázku 2.1). Každý participant má soubor stavů a aktivit, podobných stavům a přechodům v Mealyho automatu.

- **Stav** - Představuje stav (*state*), ve kterém se participant může během procesu nacházet (bílé obdélníky na obrázku 2.1). Stavy jsou hlavní součásti každého participanta. Jednotlivé stavy jsou propojeny šipkami (přechody/transitions). Pokud vede přechod ze stavu A do stavu B, znamená to, že participant, který je ve stavu A může pokračovat do stavu B. Každý participant má nejvíce jeden počáteční (v tom začíná daný proces) a minimálně jeden koncový stav (v tom proces ukončí).

- **Přechod** - Pomocí přechodu mezi stavem A a B lze (po splnění daných podmínek) pokračovat v procesu ze stavu A do stavu B.

- **Aktivita** - Může být na přechodu mezi stavy a popisuje co se děje během přechodu z jednoho stavu do druhého. Znázorňuje například potřebu něco udělat nebo komunikovat s jiným participantem. (elipsy na obrázku 2.1). Reprezentuje akci, která produkuje výstup nebo jen činnost, kterou je potřeba vykonat než je možné se přesunout do dalšího stavu.

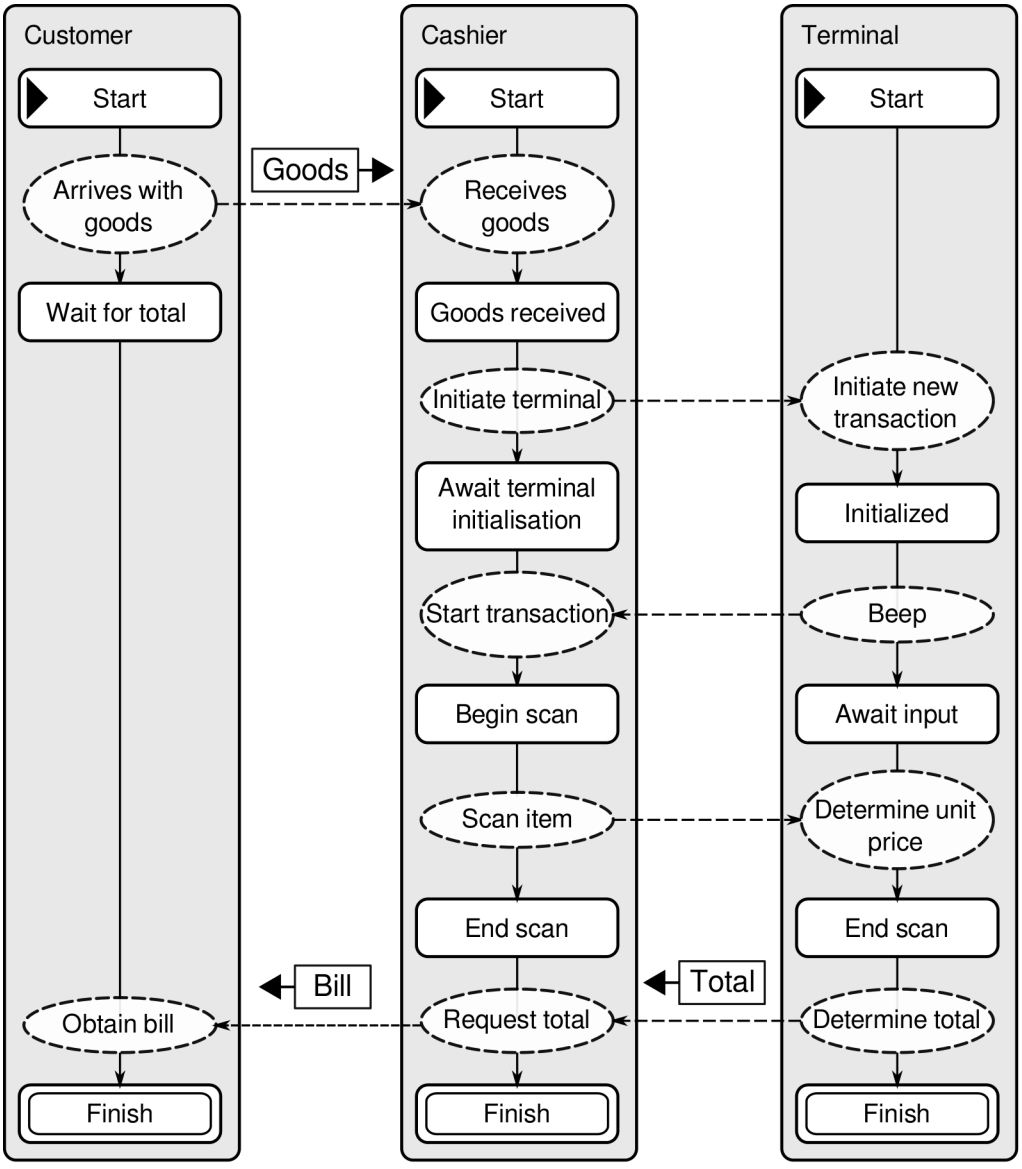
- **Komunikace** - Spojuje dvě aktivity mezi různými participanty. Vyjadřuje nutnost provést požadovaný úkon vůči druhému participantovi. Může mít pouze informační charakter (předej informaci, zavolej) nebo může obsahovat přenos fyzických věcí. Například předej peníze, převezmi zboží...

- **Přenos** - Slouží pro popis komunikace mezi dvěma participanty. Pomocí přenosu lze například popsat přenos materiálu, ale také třeba objednávku, potvrzení, platbu...

- **Podmínky** - Přechody mezi jednotlivými stavy mohou být omezeny podmínkami (*conditions*). Pokud vychází ze stavu více různých přechodů, mohou být stanoveny různé podmínky pro průchod procesem. Participant smí pokračovat v činnosti přes tento přechod pouze v případě, že dané podmínky splní. Více informací v sekci 3.8.

  Podmínky jsou definovány pomocí **logického výrazu**. Logický výraz je buď **formule** anebo **proměnná**. Proměnná symbolizuje jeden přechod do/ze stavu. Formule se skládá z **operátoru** a množiny výrazů. Operátory se vztahují vždy k výrazům ve formuli, a jsou trojího typu: **ANY** - obdoba logického *OR* pro více proměnných - stačí aby byla splněna jedna z podmínek a výraz je vyhodnocen jako pravdivý. **ALL** - obdoba logického *AND* pro více proměnných - musí být splněny všechny podmínky aby byl výraz vyhodnocen jako pravdivý. **ONE** - je obdoba *XORu*, kde

je výraz ohodnocen pravdivě pouze v případě, že je splněna právě jedna podmínka.



Obrázek 2.1: Příklad Object Relation Diagramu [1]

2.1.3 Využití

- Tato metoda byla úspěšně použita členy CCMi [10] v Deloitte Advisory Czech Republic, ve velkých projektech pro Českou Poštu, energetický průmysl, telekomunikační firmy, aerolinie a další klienty.
- Modelování business procesů na FIT ČVUT.
- Modely procesů při povodních. Projekt NAKI „Protecting Cultural Heritage from Floods“: projekt financovaný ministerstvem kultury České republiky. [10]

[10] `http://ccmi.fit.cvut.cz/methodologies/borm/`

Kapitola 3

Analýza a návrh

3.1 Generování reportů

Report je v podstatě textové shrnutí toho, co mají jednotliví účastníci procesu dělat, aby splnili procesem definovaný úkol. Například skladník pak nemusí hledět do složitého diagramu, ale dostane na papíře vytištěný slovní popis, jak má postupovat v té které situaci.

3.1.1 Příklad využití reportu

Informační systém pro odevzdávání závěrečných prací vyžaduje účast několika osob. Nejprve vedoucí práce musí vložit zadání závěrečné práce do systému. Poté si ho vybere a zarezervuje student nebo je mu přímo vedoucím nabídnuto. Student může mít nezávazně přihlášeno více témat, dokud některé finálně nepotvrdí. Dále probíhá několik různých kol úprav a potvrzování mezi studentem a vedoucím práce. V průběhu tohoto procesu je ještě možné zadání vracet k úpravám a připomínkám či ho úplně zrušit a vybrat téma nové. Nakonec po shodě na zadání a specifikaci tématu oběma stranami je potřeba, aby finální podobu zadání schválil (nebo zamítl) ještě vedoucí katedry a poté děkan fakulty.

Tento systém je poměrně složitý a nepřehledný. Zvláště, když procesní diagram čítá celkem desítky různých entit, přechodů a podmínek. V tomto případě by například bylo vhodné, kdyby každý ze zúčastněných měl dopředu jasně dané, **co musí kdy provést za činnost a jaké situace mohou nastat**, jelikož vyčíst to například z velice komplikovaného diagramu je časově náročné. Následovat kroky v textovém reportu je mnohem snazší nežli vizuálně simulovat průchod složitým grafem a vyhodnocovat průběžně různé kombinace a možnosti podmíněných postupů během procesu.

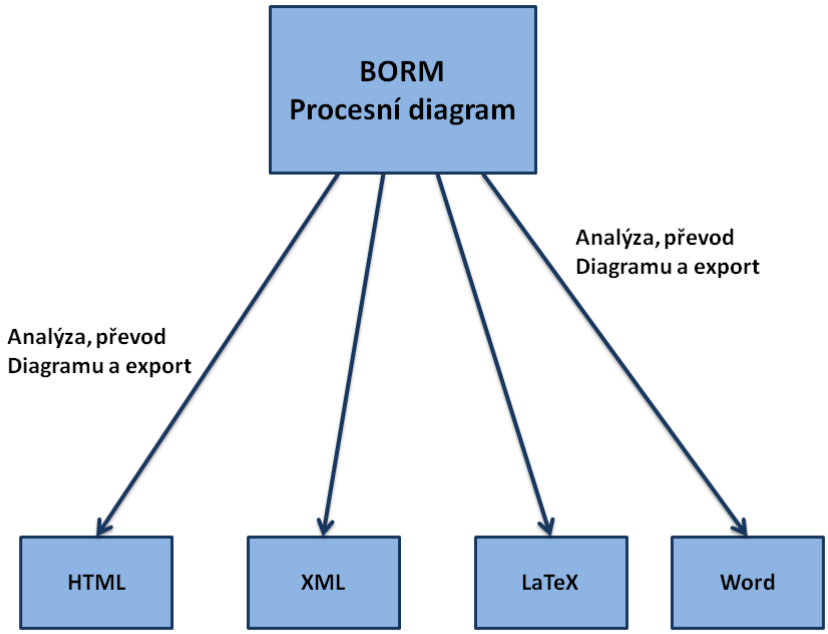
3.1.2 Požadavky na reporty

Textový report by měl být co nejpřehlednější a pro jednotlivé účastníky snadno pochopitelný. V bakalářské práci se proto zaměřuji na přehledné vyjádření vztahů z 2D diagramu do prostého textu. Je potřeba srozumitelně znázornit například větvení procesu v závislosti na splnění či nesplnění vstupních a výstupních podmínek, komunikaci s ostatními participanty nebo různé propojení stavů a aktivit v diagramu.

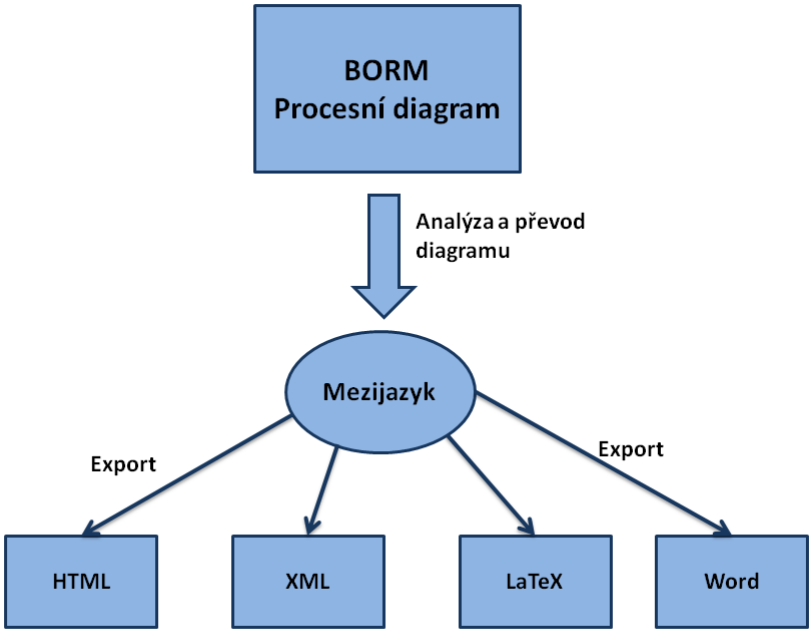
3.1.3 Postup

Diagram je potřeba postupně projít a zanalyzovat. Po analýze se musí vygenerovat možné průchody diagramem (posloupnosti stavů a aktivit), jak může participant proces vykonat. Tento výsledek je potřeba vyexportovat do požadovaného výstupního souboru v zadaném formátu (viz obr. 3.1).

Při implementaci exportu diagramu do více textových formátů je vhodné použít „mezijazyk“ (viz obr. 3.2), navržený dostatečně obecně na to, aby byl schopný pojmout drobné změny prvků diagramu v budoucnu. Ale na druhou stranu musí být co nejjednodušší, aby byl snadný pro pochopení a použitelný pro export do dalších výstupních formátů.



Obrázek 3.1: Schéma převodu přímo z diagramu



Obrázek 3.2: Schéma převodu pomocí mezijazyka

3.2 Popis a výběr nástrojů

Reporty v rámci Bakalářské práce generuji z programu DynaCASE experimentálního nástroje vědecko-výzkumné skupiny CCMi. Tento nástroj disponuje mimo jiné i možností tvorby a úpravy BORM procesních diagramů. DynaCASE je implementován v prostředí Pharo, tudíž i program pro export je vhodné mít v tomto prostředí, aby byla co nejsnazší komunikace s modelem diagramu a uživatel mohl používat export přímo z DynaCASE.

Pro účely mezijazyka mohu využít buď některý z existujících jazyků či nástrojů, popřípadě navrhnout a implementovat nástroj vlastní pokud nebude z dostupných žádný vyhovující.

3.3 Popis mezijazyka

Hlavní požadavky na mezijazyk:

- Dokáže obsáhnout všechny potřebné informace z původního diagramu
- Jednoduchá struktura, přehledný a snadný převod do exportovaného formátu

- Dostatečně obecný, aby byl schopný pojmout drobné změny v diagramu (nových druhů entit)

Tyto požadavky splňuje nástroj Pillar, který má vhodné struktury na popis dokumentu a zároveň disponuje podporou několika výstupních textových formátů včetně LaTeX a XML. Dále je také možné dospecifikovat styl výstupního souboru pomocí konfiguračních souborů a změnou parametrů exportu.

Pro rozšíření o další formáty pro export stačí pouze přidat balíček s pravidly konverze vnitřních tříd Pillaru na požadovaný výstup.

Dostupné nástroje

- **Jazyk HTML**[11] - Tento jazyk je vhodný pro tvorbu internetových stránek a prezentací. Nabízí mnoho možností pro popis formátu a vzhledu dokumentu, které by byly více než dostatečné pro vyjádření všech částí diagramu. HTML formát podporuje i používání barev, obrázků a jiných grafických prvků. Tento jazyk je „značkovací“, to znamená, že se jednotlivé segmenty „obalují“ ve zdrojovém kódu značkami (*tagy*). Tyto značky jsou většinou párové, což ale neplatí vždy (například pro odřádkování se používá nepárový tag), tudíž by byl poměrně nepraktický převod do formátů, které značkovací nejsou nebo nepoužívají značky v párech.

- **Jazyk XML**[12] - Formát navržený pro popis dat a jejich struktury. Využívá také především párové značky jako HTML, ale soustředí se hlavně na popis datové hierarchie místo na popis vzhledu. Tento problém by mohl být zmírněn definicí vlastních tagů, což XML standard umožňuje. Bohužel je XML struktura dokumentu rovněž určena pomocí párových tagů ztěžujících převod do jazyka s jinou strukturou.

- **Markdown**[13] - Markdown je nástroj na převod prostého textu do HTML, určený pro autory internetových stránek. V tomto jazyce se k formátování textu používají místo (textových) značek speciální znaky (*, #, =, -, závorky, ...). Díky tomu je text čitelný i ve zdrojové formě, a autor ušetří čas psaním speciálního znaku místo celých tagů. Například pro vložení obrázku stačí napsat `![](cesta)` namísto `<img src="cesta" alt="" title="" />`. Markdown podporuje pouze několik základních formátovacích elementů z jazyka HTML, ale ty by pro účel jednoduše formátovaného reportu dostačovaly. Největší problém ve využití oficiálního nástroje pro zpracování souboru v Markdown syntaxi je potřeba instalace a použití programovacího jazyka Perl[14].

[11] HyperText Markup Language (`http://www.w3.org/html/`)
[12] EXtensible Markup Language (`http://www.w3schools.com/xml/`)
[13] Markdown (`http://daringfireball.net/projects/markdown/`)
[14] Perl - (`https://www.perl.org/`)

- **Nástroj Pillar**[15] - Pillar využívá značkovací jazyk, který se svou syntaxí podobá Markdownu a místo tagů se formát textu určuje také kombinací speciálních znaků. Oproti Markdownu má ovšem pro naše potřeby několik významných výhod.

  - **Integrace v prosředí Pharo**: Nástroj Pillar lze používat více způsoby. Jednak lze volat z příkazové řádky pro export konkrétních souborů do požadovaného formátu, ale lze s ním pracovat i přímo v prostředí Phara. Tento způsob se velmi hodí, jelikož všechny ostatní programy jsou také ve Pharu, tudíž spolu navzájem mohou bez problému komunikovat. Pomocí metod a rozhraní lze s Pillarem interagovat přímo v rámci programu a předávat mu data například přímo jako proměnné nebo jako textové řetězce.
  - **Nativní podpora exportu**: Pillar je primárně určen pro psaní a generování dokumentace a knih [11]. Nativně tedy podporuje export do běžných textových formátů jako HTML, LaTeX a Markdown, což z Pillaru dělá velice vhodný nástroj pro využití jako mezistupně během generování textového reportu z procesního diagramu.
  - **Vnitřní reprezentace**: Pillar interně uchovává strukturu dokumentu ve formě objektů, které popisují jednotlivé části dokumentu. Různé nadpisy, odkazy, formát písma a odstavce.... Pro výsledný export je tak pouze potřeba definovat jak se který objekt má interpretovat ve výsledném souboru.
  - **Šablony**: Díky konfiguračním souborům a šablonám lze poměrně snadno měnit vzhled výstupního souboru, přidávat nové výrazy a jejich překlady do výstupních formátů. Navíc je také možné doimplementovat balíčky tříd pro export do dalších požadovaných formátů.

- **Vlastní nástroj** - Poslední možností je navrhnout vlastní mezijazyk, který bude mít všechny požadované vlastnosti a k němu navrhnout a implementovat nástroj, který bude schopný tento jazyk zpracovat a převést na požadovaný formát.

Jelikož námi požadované vlastnosti a vhodné podmínky již poskytuje vyvíjený nástroj Pillar, je nejefektivnější možnost tento nástroj pro naše potřeby využít. Splňuje jak požadavky na dostatečnou obecnost pro popis dokumentu, tak zároveň umožňuje poměrně snadný export do dalších formátů, jako jsou LaTex či HTML. Tento nástroj je také v případě potřeby dále modifikovatelný pomocí konfiguračních souborů.

[15] Pillar (`http://smalltalkhub.com/#!/ Pier/Pillar`)

3.4 Převod modelu do mezijazyka

Pro přehlednost výsledného dokumentu je vhodné vygenerovat oddělené části pro jednotlivé participanty procesu zvlášť. Každému participantovi vypsat posloupnost kroků, které musí v rámci procesu provést a s kým musí jakým způsobem komunikovat.

Nástroj Pillar umožňuje vytváření obecné formy dokumentu v mezijazyce jak za pomoci textové reprezentace a speciálních značek, tak také využitím přímo vestavěných tříd a metod.

První varianta spočívá v psaní dokumentu formou prostého textu upraveného o značky Pillaru, poté zpracování tohoto textu Pillarem, který zadaný text rozparsuje a vytvoří si vnitřní strukturu objektů a vazeb.

Druhá možnost, která je využita v rámci této práce, využívá přímého přístupu ke struktuře Pillar dokumentu v objektové podobě. Místo psaní prostého textu se vytváří přímo objekty Pillaru a vkládají se do výsledného Pillar dokumentu. Po dokončení zpracování celého diagramu vznikne kompletní dokument, který má obecnou strukturu určující pouze typy položek a jejich uspořádání, zcela nezávisle na výsledném výstupním formátu.

3.5 Export z mezijazyka do výsledného formátu

Pillar uchovává obecný formát dokumentu jako soubor objektů. Tento obecný dokument složený z objektů vznikne jak přímou prací se třídami a metodami Pillaru, tak zpracováním prostého textu s formátováním pomocí značek pro Pillar. Pro export do výsledného formátu tedy není rozdíl v tom, jakým způsobem byl náš obecný dokument vytvořen.

Samotný export se poté provádí použitím požadovaného balíčku tříd pro konkrétní výstupní jazyk. Tyto balíčky a jejich nastavení lze (obdobně jako většinu věcí v prostředí Pharo) dle potřeby měnit. Výstup je možné zapsat do souboru nebo na obrazovku.

3.6 Modelová reprezentace

Před samotným začátkem převodu je potřeba nejprve celý diagram zanalyzovat a vybrat v něm všechny možnosti průchodu procesem (postupy k jeho splnění).

Diagram se skládá z jednotlivých participantů (viz sekce 2.1.2). Každý participant je v modelu reprezentován objektem s určitými atributy, který obsahuje všechny své stavy a aktivity. Dále jsou na jednotlivé aktivity navázány komunikace a přenosy. Stavy a aktivity jsou navzájem propojeny pomocí přechodů. Stavy dále mají své vstupní a výstupní podmínky pro vyhodnocení možnosti průchodu procesem.

3.7 Input/Output conditions

Někdy může být postup v procesu podmíněn předchozím provedením nějaké akce, například nelze vystavit určité léky pacientovi, pokud nepředloží platný recept od lékaře. V jiné situaci je zase potřeba specifikovat možnosti dalšího postupu procesem a vybrat validní kombinace přechodů do dalších stavů a aktivit. Například pro identifikaci osoby je potřeba zkontrolovat občanský průkaz, ale je možné k identifikaci využít například i cestovní pas dotyčného nebo jiný doklad.

Pro definici těchto a dalších požadavků na podmíněný postup procesem se používají vstupní a výstupní podmínky resp. *input and output conditions*.

Vstupní podmínky (*Input conditions*) stavu definují, jaká kombinace vstupů do tohoto stavu musí být splněna, aby bylo možno do stavu vstoupit. Například recepční v hotelu musí nejprve ověřit totožnost zákazníka a převzít platbu za pobyt než mu může předat klíče od jeho pokoje. Pokud se zákazník věrohodně neprokáže nebo nezaplatí požadovanou částku, tak mu nemohou být klíče předány.

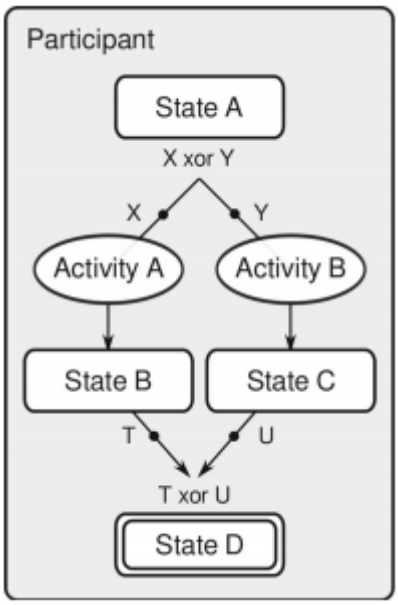
Výstupní podmínky (*Output conditions*) stavu definují, jaká kombinace výstupů z tohoto stavu musí být splněna, aby bylo možné v procesu pokračovat dále. Například zaměstnanec ve firmě si musí na začátku roku vybrat, zda chce dostávat zaměstnanecké benefity ve formě stravenek anebo permanentky do místního FIT-centra. V takovém případě si bude muset vybrat právě jednu z nabízených z možností.

Vstupní a výstupní podmínky jsou popsány pomocí **výrazů v Booleově algebře** a uloženy přímo ve stavech. Vstupní podmínky jsou popsány výrazem, jehož proměnné jsou přechody končící v tomto uzlu. Znamená to, že proces nepokročí do tohoto stavu, dokud nebudou splněny vstupní podmínky, to znamená nebude výraz vyhodnocen jako pravda. Výstupní podmínky nadruhou stranu pomocí Booleovského výrazu určují, do jakých kombinací výstupů se může proces z daného stavu rozvětvit.

3.8 Vyhodnocení podmínek postupu diagramem

Při vyhodnocování podmínek průchodu využívá program rozklad logických výrazů (podmíněných přechodů) na **disjunktní normální forma** (DNF), kde každý z disjunktních výrazů interpretuje jeden možný průchod grafem (= průchod procesem). Rozklad na DNF lze provést přes pravdivostní tabulku, kde jako proměnné slouží vstupní resp. výstupní přechody. Výsledné kombinace možných průchodů lze pak dostat jako řádky, kde je daná podmínka splněna. Průchod pak bude pokračovat všemi přechody, které jsou na daném řádku ohodnoceny jako pravdivé.

Každá varianta průchodu poté dostane svůj identifikátor a čtenář se rozhodne podle aktuálního stavu/splnění podmínek, jak dále pokračovat (který



Obrázek 3.3: Příklad vstupních a výstupních podmínek [2]

identifikátor vyhledat v textu a od toho pokračovat dále).

3.9 Konverze do textové formy

Díky rozkladu formule na DNF lze postihnout všechny možné kombinace průchodů. Každý jeden takový průchod bude dle potřeby v textu označen značkou, aby bylo možné se na něj z rozvětvujícího se stavu odkázat. Zároveň to umožní širší větvení (za cenu opakování některých textových pasáží procesu), než by šlo realizovat například pomocí rozepisování větví do sloupečků v tabulce, kde by každý sloupeček znázorňoval jeden průchod. Tento způsob je velice přehledný pro malý počet možností, které se již dále nevětví (omezení počtu sloupečků šířkou stránky).

3.10 Orientace v textu

Text je ve výsledném dokumentu členěn chronologicky tak, jak by měl participant procházet po řadě danými stavy v procesu (viz příloha B.2). Jednotlivé stavy a rozhodovací úseky se označují identifikátorem, na který se lze posléze v případě potřeby odkázat v textu. Tyto identifikátory stačí mít pouze v uzlech, které se prochází víckrát, což algoritmus pozná během analýzy diagramu. Větvení procesu lze v textu vyjádřit například jako tabulku, kde sloupečky budou jednotlivé průchody. Takové znázornění je ale značně limitováno počtem větvení, které se vedle sebe vejdou na stránku. Například pro možnosti ANO/NE

by tabulka stačila, v levém sloupečku by byl průchod pro možnost ANO, v pravém sloupečku pro NE. Pro více možností (například při výběru služby z několika možných) už začne být tabulka poměrně komplikovaná. Proto se různé průchody skládají za sebe a je na ně odkazováno podle splnění všech potřebných podmínek.

Kapitola 4

Realizace

4.1 Použité technologie

Samotné diagramy jsou namodelovány pomocí nástroje DynaCASE, vyvíjeným výzkumnou skupinou CCMi a studenty na Fakultě informačních technologií. DynaCASE, v současné době používaný ve formě balíčků tříd v IDE Pharo, funguje na platformě Smalltalk.

Část programu, která zajišťuje zpracování a převody diagramu je rovněž implementována do prostředí Pharo v jazyce Smalltalk. Díky tomu není potřeba využívat žádné další externí nástroje na komunikaci a přenos dat mezi DynaCASE a exportovacím nástrojem. V rámci prostředí spolu komunikují jako klasické objekty v programu.

Mezijazyk a část pro převod mezijazyka do požadovaného výstupního formátu jsou obsluhovány nástrojem Pillar, který lze také využívat přímo z prostředí Pharo. Tím odpadá nutnost zajišťovat externí komunikaci mezi různými nástroji napříč platformami.

Prostředí Pharo lze provozovat jak na systémech MS Windows, tak na MAC OSX či GNU/Linux.

4.2 Programové součásti práce

Nad rámec této bakalářské práce byl kromě samotného exportování doimplementován balíček pro správu BORM diagramů do DynaCASE, nad kterým operuje exportovací nástroj vyvíjený v této bakalářské práci. Na rozdíl od ostatních metod v DynaCASE, pro BORM není v době psaní této práce vytvořeno grafické rozhraní. Exportovací nástroj využívá z diagramu ale pouze jeho vnitřní reprezentaci, která je na grafickém znázornění zcela nezávislá. Tudíž bude snadno obsluhovatelný i z grafického prostředí pouze zavoláním metody pro export a předáním diagramu k exportu. Během vývoje je tedy diagram tvořen přímo vytvářením objektů a voláním metod potřebných tříd z konzole Phara.

Dále byl implementován soubor tříd obstarávajících analýzu a zpracování diagramu s následným převodem do mezijazyka Pillar. Uživatel programu použije pouze tento exportovací nástroj, tudíž se balíček musí postarat o vytvoření Pillar dokumentu a jeho následnou obsluhu spojenou s exportem do výstupního formátu.

Poslední součást je mezijazyk zprostředkovaný nástrojem Pillar rovněž pod IDE Pharo. Tento nástroj umožňuje vysokou modularitu díky možnosti přidáváních dalších exportovacích formátů a navíc je velice přizpůsobitelný potřebám uživatele. Snadno lze v konfiguračních souborech upravit například vzhled výsledných elementů exportovaného dokumentu nebo přepsat část šablony.

4.3 Struktura programu

Části této bakalářské práce jsou implementovány v balíčku Borm-Model, a jsou rozděleny do čtyř základních skupin podle oblasti, kterou řeší. Models, Conditions, Exporter a Tests. Zde uvádím, jakých částí programu se tyto skupiny týkají.

V další sekci je poté blíže popsána implementační stránka nejpodstatnějších oblastí programu. Důležitá zejména pro případné další rozšíření či práci s exportovacím nástrojem.

1. **Models** - Tato skupina obsahuje třídy popisující samotný BORM procesní diagram. Jsou zde třídy definující participanta, stavy, komunikace, přechody...

2. **Conditions** - Zde jsou části programu popisující logiku a podmínky přechodů mezi stavy. Například proměnné, pravdivostní tabulky, operátory, logické výrazy a způsoby výpočtu.

3. **Exporter** - Exportování diagramu a jeho převod do mezijazyka Pillaru zajišťují třídy v této skupině. Jednak je tu procházení diagramu, rozhodování se podle input a output conditions, ale také se zde obsluhuje a zařizuje samotný převod do mezijazyka a poté i export do výsledného formátu.

4. **Tests** - Veškeré testy i testovací soubory pro ověření funkčnosti se nacházejí pod touto záložkou. Také tu jsou ukázkové diagramy a jiné objekty, na kterých se dá demonstrovat nebo ověřovat tvorba a exportování BORM diagramů.

Jednotlivé skupiny spolu mohou bez problémů komunikovat a slouží spíše pro přehlednost v rámci prostředí. Není potřeba nijak explicitně zadávat cestu do jiné skupiny nebo jiného balíčku ani přenastavovat namespace, jako je tomu u mnoha jiných jazyků.

Program Pillar je uložen ve zvláštních balíčcích, a komunikují s ním pouze třídy ze skupiny Exporter během generování obecného dokumentu a zařizování exportu do souboru.

4.4 Implementace

- **Model** je složen z poměrně jednoduchých tříd s intuitivní hierarchií. Diagram má například kolekci participantů, participant má kolekci vnitřních stavů, každý stav má kolekci vstupních, výstupních přechodů a patřičných podmínek...

- **Exporter** obsahuje jednu třídu na zpracování diagramu a druhou na práci s exportem do výstupního souboru. Tyto třídy prochází a analyzují graf (procesní diagram). Postupně se analyzuje každý participant zvlášť. Vždy se vybere iniciální (počáteční) uzel, který se pozná buď podle toho, že má tuto vlastnost zadanou (u stavů) anebo nemá žádné vstupní přechody (u aktivit).

- **Průchod grafem** probíhá pomocí algoritmu prohledávání do hloubky (popsaným např. zde [12] str. 78) s některými úpravami pro zohlednění podmínek podle článku [2].

  Zjednodušeně lze popsat postup tak, že se během průchodu grafem nastavuje jednotlivým uzlům a přechodům příznak, zda byly navštívené, a jestli to byla **pozitivní** či **negativní návštěva** (*lze se do prvku dostat během průchodu či nelze*). Klasické procházení označuje prvky jako pozitivně navštívené, negativní procházení jako negativně navštívené.

  - **Kontrola vstupních podmínek** probíhá v této verzi tak, že stačí, aby byly všechny příchozí přechody alespoň jednou navštíveny. Pokud byla minimálně jedna z cest navštívena pozitivně, znamená to, že se lze do tohoto uzlu dostat a pokračuje se dál. Pokud jsou všechny příchozí hrany negativně navštívené, do uzlu se nelze v tomto průchodu dostat, a všechny jeho výstupy jsou negativně prohledány (protože ani do navazujících uzlů se nelze dostat z tohoto uzlu).
  - **Vypsání uzlu** se provede pouze v případě, že jsou všechny vstupní přechody navštívené, a alespoň jeden z nich pozitivně navštíven. Provádí se zde například vypsání obsahu uzlu, jeho komunikací, výpočet podmínek...
  - **Další průchody** jsou určeny výstupními podmínkami. V uzlech, kde je více výstupních přechodů se vyhodnotí výstupní podmínky a vygeneruje se pravdivostní tabulka. V této tabulce symbolizuje každá proměnná jeden výstupní přechod. Na základě této tabulky

se poté rekurzivně prochází různé kombinace výstupních přechodů. Pokud je v tabulce možný průchod pro určité ohodnocení výstupních přechodů, tato kombinace se zapamatuje. nejprve se provede negativní průchod pro přechody, kudy podmínka průchod nedovoluje a poté klasický průchod pro přechody s ohodnocením *true*.

Negativní průchody se provádí první proto, aby se poté zbytečně nečekalo s vyhodnocováním uzlů během klasického průchodu. Čekat s vyhodnocením se bude pouze v případě, že do daného uzlu opravdu ještě vede nějaká další validní cesta.

Ukázka diagramu a výsledného exportu lze nalézt v příloze této práceB.1.

4.5 Programátorská příručka

tato příručka popisuje pouze postup pro použití samotného exportovacího nástroje, nikoli postup pro tvorbu samotných BORM diagramů.

Pro správné fungování exportovacího nástroje vyvíjeného v rámci této práce je potřeba mít nahrané v prostředí Pharo tyto balíčky tříd: **Models**, kde je definována struktura diagramů. **Conditions**, kde jsou popsány logické výrazy a práce s nimi. **Exporter**, kde je implementace logiky a nástrojů pro export (viz 4.4). Navíc jsou potřeba ještě balíčky pro Pillar, jejichž stažení je popsáno v návodu níže.

1. **Načtení Pillaru** do prostředí Pharo:

   Spuštění následujícího programu provede připojení k úložišti zdrojových kódů a stažení nástroje Pillar, který je potřeba pro správné fungování exportu diagramů.

   ```
   Gofer new
       smalltalkhubUser: 'Pier' project: 'Pillar';

       configuration;

       load.

   (Smalltalk globals at: \#ConfigurationOfPillar) load.
   ```

2. **Vytvoření instance** exportovacího nástroje:

   tento kód vytvoří novou instanci exportovacího nástroje s názvem *"myExporter"*, která nám později umožní náš diagram převést do požadovaného textového formátu.

   ```
   myExporter := BormParser new.
   ```

3. **Nastavení cesty**:

   pro nastavení cesty, kam se má výsledný exportovaný soubor uložit je potřeba tuto cestu specifikovat pomocí následující metody, kde je cesta parametr *aPath* (část za dvojtečkou).

   ```
   myExporter setPath: aPath
   ```

   Defaultně se exportované soubory ukládají do stejné složky, ve které je spouštěný program. Pozor na to, že při exportování nového diagramu se starý soubor se stejným jménem přepíše!

4. **Export diagramu**:

   Takto vytvořenému objektu myExporter předáme parametrem (část za dvojtečkou) diagram, který chceme exportovat.

   ```
   myExporter n_exportDiagram: aDiagramToExport.
   ```

4.6 Přidání dalších formátů

Pokud je potřeba exportovat do jiných, formátů než LaTeX a HTML, je možné upravit metody exportovacího nástroje tak, aby volaly jiné části Pillaru (například pro formát Markdown). V ostatních případech, které nejsou defaultně v nástroji Pillar, je možné přidat vlastní rozšíření (šablonu) přímo do Pillaru. S takovým rozšířením lze poté pracovat stejně jako s exportem do HTML a LaTeXu.

4.7 Oblasti modelu nepokryté touto prací

Exportovací nástroj zatím neumí správně operovat s obousměrnými datovými toky v rámci jedné komunikace a nepracuje s ověřením vstupních podmínek a verifikací jejich korektního zadání. Tomuto tématu se věnuje jiná bakalářská práce, tudíž zde již počítám s tím, že zadané cesty a podmínky jsou korektní. Pokud se lze do uzlu dostat díky výstupním podmínkám předchozího uzlu, tak předpokládám shodu se vstupními podmínkami tohoto uzlu a již je zpětně neověřuji.

Kapitola 5

Testování

Testování se zaměřuje na statickou analýzu kódu a ověření funkčnosti programu.

5.1 Statická analýza

Syntaktická správnost:

Pro kontrolu syntaxe má Pharo několik kontrolních mechanismů, které pomáhají programátorovi udržovat kód pokud možno bez syntaktických chyb.

- **Okamžitá kontrola** - Prostředí Pharo automaticky kontroluje syntaxi již při psaní zdrojového kódu a případné chyby se projeví nesprávným zvýrazněním dotyčné části. Například při špatném ukončení volání funkce se zbytek kódu zvýrazní stejnou barvou jako parametry funkce. Z toho lze usoudit, že bude celý zbytek kódu interpretován jako parametr funkce, což je obvykle špatně.

- **Kontrola při ukládání** - Před samotným opuštěním módu pro editaci zdrojového kódu a uložením změn se vždy provede syntaktická analýza celé oblasti, kde byly prováděny změny (například celé metody, rozhraní třídy...). V případě nalezení chyby se změna neuloží a zastaví se opuštění metody. V prostředí, obvykle přímo na daném řádku, se zobrazí, k jaké chybě došlo. Dokud tedy programátor nemá napsaný kód syntakticky správně, nemůže ho ani uložit.

  Úprava kódu a sémantika:

  podle kapitoly 6, sekce o verifikaci objektového návrhu v [13] jsem se zaměřil na kontrolu některých prohřešků (kapitola 6.1.1).

- **Názvy zpráv a metod** by měly být pojmenovány podle akce, kterou vykonávají a ne podle způsobu, jakým ji vykonávají. Zejména proto,

že pro odesílatele zprávy je důležitý především výsledek akce a nikoli vnitřní algoritmus volané metody.

Splnění tohoto požadavku bylo dosaženo díky postupu psaní kódu, kdy jsem nejprve vytvářel metody a vhodně je pojmenovával podle očekávané akce, kterou vykonají. Konkrétní implementaci samotných metod a tříd jsem provedl až následně, tudíž názvy nebyly odvozeny od vnitřního algoritmu metod.

- **Pojmenování proměnných, parametrů, metod** by mělo vycházet z jejich role. Dočasné a instanční proměnné by měly být označeny podle nejobecnější třídy, která je očekávána. Například dodržování zvyku nazývat parametry *aString* v případě očekávání řetězce, *anInteger* zase pro celočíselný parametr. Toto pravidlo jsem se snažil v rámci bakalářské práce dodržovat.

- **Délka metod a duplikace kódu** - Obecně by metody měly mít maximálně kolem deseti řádků kódu, když nepočítáme prázdné řádky, komentáře a závorky na samostatných řádcích. Rozsáhlejší metody ubírají programu na přehlednosti a obvykle značí provádění více činností v jedné metodě než je nutné. Výjimku tvoří metody, které dále nelze dělit, jako například některé metody v rámci testů, které vytvářejí celé diagramy a každou část musí zvlášť nastavit.

  Délku metod kontroluje samotné Pharo, které na barevné stupnici hodnotí počet znaků v dané metodě.

  Místa, kde se vyskytovala duplikace kódu, jsem nahradil novou metodou. Tím se opět zvýšila přehlednost kódu a zjednodušily případné úpravy. Změny pak stačí provádět pouze v jedné metodě místo hledání všech výskytů, které byly vytvořeny.

5.2 Funkční testy

V balíčku „**Tests**“ jsou uloženy všechny funkční testy. Jak pro část s implementací BORM diagramu, tak pro ověření správnosti fungování samotného exportu.

Funkční testování zahrnuje tyto základní prvky:

- **Vytváření instancí** používaných objektů s předem určenými vlastnostmi a vazbami. Například se zde vytváří jednotlivé stavy, aktivity a komunikace. Participanti a celé diagramy s různými nastaveními...

- **Ověření** předpokládaného **chování** a reakcí na různé zprávy. Testování zda vytvoření komunikace mezi aktivitami tyto aktivity skutečně propojí, zda se správně identifikuje startovní uzel atp.

- **Logické výrazy** a jejich vyhodnocování v rámci provádění exportu. Tvorba diagramů, kde mají participanti více možností přechodů mezi stavy. Kontrola tvorby pravdivostních tabulek a správných hodnot.
- **Export** a jeho vizualizace jsou testy, jejichž správný výstup se musí hodnotit ručně (ověřit, zda export vypadá tak jak jsme chtěli například). Provede se na vybraných diagramech k tomuto účelu určených a výsledek se zkontroluje v patřičném výstupním souboru.

5.3 Výkonnostní testy

Základní diagramy obsahující až tři participanty a řádově desítky stavů se vyhodnocují téměř okamžitě. Nejnáročnější je výpočet pravdivostních tabulek, který dosahuje složitosti $O(n * 2^n)$, kde „n“ je počet přechodů (proměnných) v tabulce. Pro každou proměnnou se musí vygenerovat 2^n pravdivostních hodnot. To znamená, že pro stavy s deseti a méně přechody se jedná řádově maximálně o tisíce operací. Pro větší diagramy nebyl program testován z důvodu absence grafického rozhraní, ve kterém by je bylo možno snáze vytvořit.

Závěr

Cílem této práce bylo navrhnout a implementovat generátor textových reportů z procesních diagramů vytvořených metodou BORM. Během analýzy daného problému byly vybrány vhodné nástroje pro převod procesních diagramů do textové podoby i nástroj Pillar pro reprezentaci obecného mezijazyka, který usnadňuje export do dalších výstupních formátů.

V implementační části byl nejprve naprogramován systém logických výrazů a jejich vyhodnocování. S těmito prvky již šlo BORM diagram správně exportovat do požadovaného výstupního formátu, takže mohl být naprogramován i samotný nástroj pro export.

Jelikož nebyl nástroj pro tvorbu BORM diagramů v DynaCASE zcela hotov, musel jsem se před začátkem samotné práce důkladně seznámit s návrhem implementace BORMu a potřebné části doprogramovat. To mě sice stálo mnoho času navíc, na druhou stranu mi to ale pomohlo lépe pochopit vnitřní strukturu diagramů a poté efektivněji navrhnout samotný nástroj pro export.

Pro návrh algoritmu zpracování diagramů jsem využil znalosti z předmětu *Grafové algoritmy a základy teorie složitosti* (BI-GRA). Dále byly velmi přínosné konzultace s vedoucím bakalářské práce Mgr. Martinem Podlouckým a některé postupy popsané v článku o BORMu [2].

S jazykem Smalltalk a prostředím Pharo jsem se setkal v předmětu *Objektové modelování* (BI-OMO), díky čemuž jsem si ho vybral pro implementační část této práce. Dále jsem využil znalosti z tohoto předmětu k návrhu celého nástroje a co nejvhodnějšího rozdělení programu na jednotlivé části.

Důležitou součást práce pak tvořila oblast s logickými výrazy, jejich vytvářením a vyhodnocováním. Zde jsem pro změnu uplatnil poznatky z *Matematické logiky* (BI-MLO)

Výhled do budoucna

Nyní nástroj pro export funguje jako samostatný program, kterému se předá procesní diagram, a výsledkem je vygenerovaný soubor s textovým výstupem. Zatím ale neoperuje s některými součástmi BORM diagramu, jako jsou například obousměrné datové toky v rámci komunikace či verifikace vstupních podmínek. To jsou témata, která mohou být zpracována v rámci dalších školních prací.

Jakmile se dokončí grafické prostředí pro tvorbu BORM diagramů v nástroji DynaCASE, bude možné exportovat diagram přímo z editoru. Postačí pouze poslat do exportovacího nástroje celý diagram s případným nastavením, což obnáší zavolání jediné metody.

V další verzi by mohl mít uživatel různé možnosti jak upravit výsledek exportu. Například editaci šablon (barvy, velikost písma a font), vložení vlastního loga na začátek dokumentu...

Dále by mohl přibýt náhled výsledného dokumentu přímo z editoru diagramů. A samozřejmě v případě zájmu podpora pro další výstupní formáty.

Osobní přínos

Během této práce jsem v praxi uplatnil mnoho znalostí z celé škály předmětů, které jsem ve škole absolvoval. Tyto znalosti jsem se naučil kombinovat a aplikovat na konkrétní dílčí problémy v rámci většího projektu, což je pro mě důležitá zkušenost.

Také jsem se naučil pracovat s novými nástroji, což považuji za další velký přínos. Například použití Pillaru jsem konzultoval přímo s jeho tvůrcem prostřednictvím emailu.

Literatura

[1] Knott, R.; Merunka, V.; Polák, J.: Process Modeling for Object Oriented Analysis using BORM Object Behavioral Analysis. In *4th International Conference on Requirements engineering, 2000. Proceedings*, IEEE, 2000.

[2] Podloucký, M.; Pergl, R.: The Prefix Machine - A Formal Foundation for the BORM OR Diagrams Validation and Simulation. apr 2014.

[3] Barjis, J.: *The importance of business process modeling in software systems design*. Science of Computer Programming, 2008.

[4] Vejražková, Z.: *Business Process Modeling and Simulation: DEMO, BORM and BPMN*. Diplomová práce, Czech Technical University in Prague, Faculty of Information Technology, 2013.

[5] Manuel, L.; Johan, M.: *Business process modeling, simulation and design*. CRC Press, druhé vydání, 2013.

[6] Silver, B.: *BPMN Method and Style with BPMN Implementer's Guide: A structured approach for business process modeling and implementation using BPMN 2.0*. Cody-Cassidy Press, druhé vydání, oct 2011.

[7] Dietz, J. L.: *Enterprise ontology: theory and methodology*. Springer, 2006.

[8] Merunka, V.; Polák, J.: BORM Business Object Relation Modeling Popis metody se zaměřením na úvodní fáze analýzy I.S. Dostupné z: `http://www.osu.cz/prf/katedry/informatiky/aktuality/sbornik99/merunka2.html`

[9] Podloucký, M.; Pergl, R.: Towards Formal Foundations for BORM ORD Validation and Simulation. In *Proceedings of the 16th International Conference on Enterprise Information Systems*, apr 2014.

[10] BORM. apr 2015. Dostupné z: `http://ccmi.fit.cvut.cz/methodologies/borm/`

[11] Damien, C.: Pillar. apr 2015. Dostupné z: `http://smalltalkhub.com/#!/~Pier/Pillar`

[12] Kolář, J.: *Teoretická informatika*. Praha: Česká informatická společnost, druhé vydání, 2000.

[13] Balda, M.: *Portál pro optimalizaci BORM procesů*. Bakalářská práce, České vysoké učené technické v Praze, Fakulta informačních technologií, 2013.

Příloha **A**

Seznam použitých zkratek

BORM Business Objects Relation Modelling

BPM Business process modeling

BPMN Business Process Model and Notation

CCMi Centre for Conceptual Modelling and Implementations

DEMO Design & Engineering Methodology for Organizations

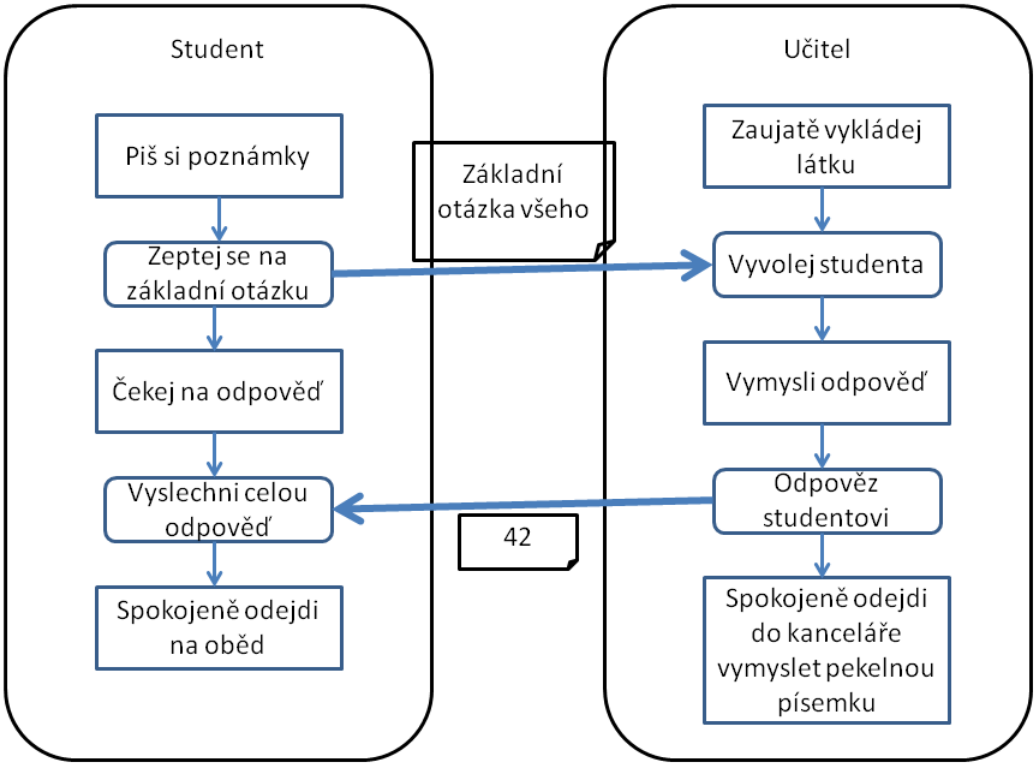
DynaCASE Dynamic Computer Aided Software Engineering Tool

XML Extensible markup language

Příloha B

Přílohy

B.1 Diagram pro export



Obrázek B.1: Jednoduchý ukázkový BORM diagram pro export

B.2 Reprezentace v jazyce Pillar

```
!Export diagramu:

!!Participant: Student
!!!Pis si poznamky
pokracuj
!!!Zeptej se na zakladni otazku

odesli zpravu " '' Zakladni otazka vseho ''
" participantovi ""Ucitel, ""

pokracuj
!!!Cekej na odpoved
pokud dorazily tyto zpravy: " '' 42 ''
" od participanta ""Ucitel""
,
pokracuj
!!!Vyslechni celou odpoved
pokracuj
!!!Spokojene odejdi na obed

~~~~~~~~~~~~~~~~~~~

!!Participant: Ucitel
!!!Zaujate vykladej latku
pokud dorazily tyto zpravy: " '' Zakladni otazka vseho ''
" od participanta ""Student""
,
pokracuj
!!!Vyvolej studenta
pokracuj
!!!Vymysli odpoved
pokracuj
!!!Odpovez studentovi

odesli zpravu " '' 42 ''
" participantovi ""Student, ""

pokracuj
!!!Spokojene odejdi do kancelare vymyslet pekelnou pisemku

~~~~~~~~~~~~~~~~~~~
!Exportovani dokonceno
```

B.3 HTML verze

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>No title</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="./bootstrap.min.css" rel="stylesheet">
    <link href="./bootstrap-theme.min.css" rel="stylesheet">
    <link rel="stylesheet" href="./default.min.css">
    <script src="./highlight.min.js"></script>

    <!--[if lt IE 9]>
      <script src="./html5shiv.js"></script>
      <script src="./respond.min.js"></script>
    <![endif]-->
  </head>
  <body>

    <div class="container">
      <h1>Export diagramu:</h1><a id="Export diagramu:"></a>

<h2>1. Participant: Student</h2><a id="Participant: Student"></a>

<h3>1.1. Pis si poznamky</h3><a id="Pis si poznamky"></a><p>pokracuj</p>

<h3>1.2. Zeptej se na zakladni otazku</h3>

<a id="Zeptej se na zakladni otazku"></a>
<p>odesli zpravu &amp;quot; <em> Zakladni otazka vseho </em>
&amp;quot; participantovi <strong>Ucitel</strong></p>
<p>pokracuj</p>

<h3>1.3. Cekej na odpoved</h3>

<a id="Cekej na odpoved"></a>

<p>pokud dorazily tyto zpravy: &amp;quot; <em> 42 </em>
&amp;quot; od participanta <strong>Ucitel</strong>
,
pokracuj</p>

<h3>1.4. Vyslechni celou odpoved</h3>
```

```
<a id="Vyslechni celou odpoved"></a><p>pokracuj</p>

<h3>1.5. Spokojene odejdi na obed</h3><a id="Spokojene odejdi na obed"></a>
<p>~~~~~~~~~~~~~~~~~~~~</p>

<h2>2. Participant: Ucitel</h2>

<a id="Participant: Ucitel"></a>

<h3>2.1. Zaujate vykladej latku</h3>

<a id="Zaujate vykladej latku"></a>

<p>pokud dorazily tyto zpravy: &quot; <em> Zakladni otazka vseho </em>
&quot; od participanta <strong>Student</strong>
,
pokracuj</p>

<h3>2.2. Vyvolej studenta</h3><a id="Vyvolej studenta"></a><p>pokracuj</p>

<h3>2.3. Vymysli odpoved</h3>

<a id="Vymysli odpoved"></a><p>pokracuj</p>

<h3>2.4. Odpovez studentovi</h3><a id="Odpovez studentovi"></a>
<p>odesli zpravu &quot; <em> 42 </em>
&quot; participantovi <strong>Student</strong></p>
<p>pokracuj</p>

<h3>2.5. Spokojene odejdi do kancelare vymyslet pekelnou pisemku</h3>

<a id="Spokojene odejdi do kancelare vymyslet pekelnou pisemku"></a>
<p>~~~~~~~~~~~~~~~~~~~~</p>

<h1>Exportovani dokonceno</h1>

<a id="Exportovani dokonceno"></a>
    </div>

    <script src="https://code.jquery.com/jquery.js"></script>
    <script src="./bootstrap.min.js"></script>
    <script>hljs.initHighlightingOnLoad();</script>
  </body>
</html>
```

B.4 LaTeX verze

```
\documentclass[a4paper,10pt,twoside]{book}


\usepackage{graphicx}
\usepackage[normalem]{ulem}

\newcommand{\ct}[1]{{\sffamily\textup{#1}}}
\usepackage{listings}
\usepackage{textcomp}

\lstnewenvironment{code}[1]{%
\lstset{%
% frame=lines,
frame=single,
framerule=0pt,
mathescape=false
}
}{}

\lstnewenvironment{script}[2][defaultlabel]{%
\renewcommand{\lstlistingname}{Script}%
\lstset{
% frame=lines,
frame=single,
framerule=0pt,
mathescape=false,
name={Script},
caption={\emph{#2}},
label={scr:#1}
}
}{}
\usepackage{fixltx2e}


\usepackage[
colorlinks=true,
linkcolor=black,
urlcolor=black,
```

```
citecolor=black
]{hyperref}

\begin{document}

\part{Export diagramu:}\label{Export diagramu:}\chapter{Participant: Student}
\label{Participant: Student}\section{Pis si poznamky}\label{Pis si poznamky}
pokracuj
\section{Zeptej se na zakladni otazku}\label{Zeptej se na zakladni otazku}
odesli zpravu \symbol{34} \textit{ Zakladni otazka vseho }
\symbol{34} participantovi \textbf{Ucitel, }

pokracuj
\section{Cekej na odpoved}\label{Cekej na odpoved}
pokud dorazily tyto zpravy: \symbol{34} \textit{ 42 }
\symbol{34} od participanta \textbf{Ucitel}
,
pokracuj
\section{Vyslechni celou odpoved}\label{Vyslechni celou odpoved}
pokracuj
\section{Spokojene odejdi na obed}\label{Spokojene odejdi na obed}
\texttildelow{}
\texttildelow{}\texttildelow{}\texttildelow{}\texttildelow{}\texttildelow{}
\texttildelow{}
\texttildelow{}\texttildelow{}\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}\texttildelow{}\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}\texttildelow{}
\texttildelow{}
\chapter{Participant: Ucitel}\label{Participant: Ucitel}
\section{Zaujate vykladej latku}
\label{Zaujate vykladej latku}
pokud dorazily tyto zpravy: \symbol{34} \textit{ Zakladni otazka vseho }
\symbol{34} od participanta \textbf{Student}
,
pokracuj
\section{Vyvolej studenta}\label{Vyvolej studenta}
pokracuj
\section{Vymysli odpoved}\label{Vymysli odpoved}
pokracuj
\section{Odpovez studentovi}\label{Odpovez studentovi}
odesli zpravu \symbol{34} \textit{ 42 }
\symbol{34} participantovi \textbf{Student, }

pokracuj
```

```
\section{Spokojene odejdi do kancelare vymyslet pekelnou pisemku}
\label{Spokojene odejdi do kancelare vymyslet pekelnou pisemku}
\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}
\texttildelow{}\texttildelow{}
\texttildelow{}
\part{Exportovani dokonceno}\label{Exportovani dokonceno}

\end{document}
```

B.5 PDF verze

Part I

Export diagramu:

Chapter 1

Participant: Student

1.1 Pis si poznamky

pokracuj

1.2 Zeptej se na zakladni otazku

odesli zpravu " *Zakladni otazka vseho* " participantovi **Ucitel,**
pokracuj

1.3 Cekej na odpoved

pokud dorazily tyto zpravy: " *42* " od participanta **Ucitel** , pokracuj

1.4 Vyslechni celou odpoved

pokracuj

1.5 Spokojene odejdi na obed

~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~

Chapter 2

Participant: Ucitel

2.1 Zaujate vykladej latku

pokud dorazily tyto zpravy: " *Zakladni otazka vseho* " od participanta **Student** , pokracuj

2.2 Vyvolej studenta

pokracuj

2.3 Vymysli odpoved

pokracuj

2.4 Odpovez studentovi

odesli zpravu " *42* " participantovi **Student,**
pokracuj

2.5 Spokojene odejdi do kancelare vymyslet pekelnou pisemku

~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~

Part II

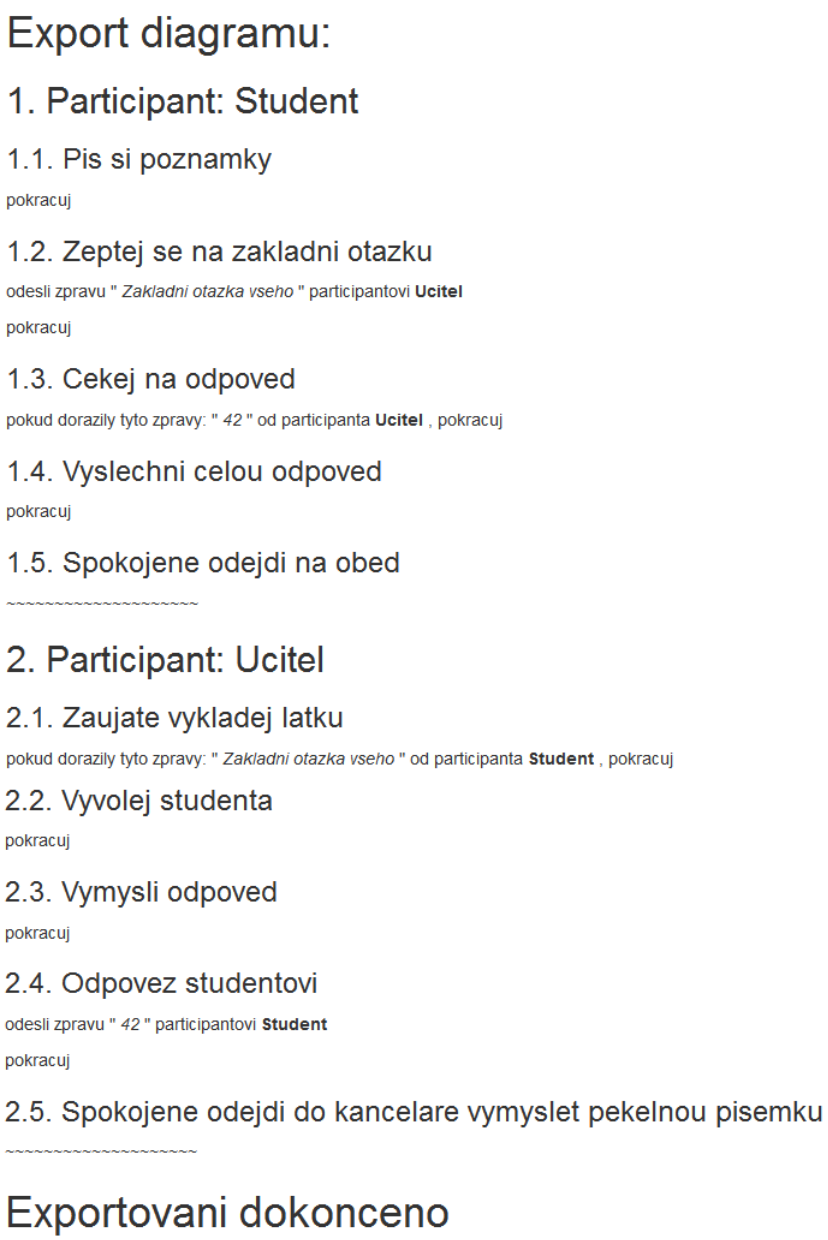
Exportovani dokonceno

B.6 Markdown verze

```
#Export diagramu:
<a name="Export diagramu:"></a>



##1\. Participant: Student
<a name="Participant: Student"></a>

###1\.1\. Pis si poznamky
<a name="Pis si poznamky"></a>pokracuj


###1\.2\. Zeptej se na zakladni otazku
<a name="Zeptej se na zakladni otazku"></a>
odesli zpravu " * Zakladni otazka vseho *
" participantovi **Ucitel, **

pokracuj


###1\.3\. Cekej na odpoved
<a name="Cekej na odpoved"></a>pokud dorazily tyto zpravy: " * 42 *
" od participanta **Ucitel**
,
pokracuj


###1\.4\. Vyslechni celou odpoved
<a name="Vyslechni celou odpoved"></a>pokracuj


###1\.5\. Spokojene odejdi na obed
<a name="Spokojene odejdi na obed"></a>
~~~~~~~~~~~~~~~~~~~~



##2\. Participant: Ucitel
<a name="Participant: Ucitel"></a>
```

```
###2\.1\. Zaujate vykladej latku
<a name="Zaujate vykladej latku"></a>
pokud dorazily tyto zpravy: " * Zakladni otazka vseho *
" od participanta **Student**
,
pokracuj


###2\.2\. Vyvolej studenta
<a name="Vyvolej studenta"></a>pokracuj


###2\.3\. Vymysli odpoved
<a name="Vymysli odpoved"></a>pokracuj


###2\.4\. Odpovez studentovi
<a name="Odpovez studentovi"></a>
odesli zpravu " * 42 *
" participantovi **Student, **

pokracuj


###2\.5\. Spokojene odejdi do kancelare vymyslet pekelnou pisemku
<a name="Spokojene odejdi do kancelare vymyslet pekelnou pisemku"></a>
~~~~~~~~~~~~~~~~~~~


#Exportovani dokonceno
<a name="Exportovani dokonceno"></a>
```

B.7 HTML verze v prohlížeči



Export diagramu:

1. Participant: Student

1.1. Pis si poznamky

pokracuj

1.2. Zeptej se na zakladni otazku

odesli zpravu " *Zakladni otazka vseho* " participantovi **Ucitel**

pokracuj

1.3. Cekej na odpoved

pokud dorazily tyto zpravy: " *42* " od participanta **Ucitel** , pokracuj

1.4. Vyslechni celou odpoved

pokracuj

1.5. Spokojene odejdi na obed

~~~~~~~~~~~~~~~~~~~~

2. Participant: Ucitel

2.1. Zaujate vykladej latku

pokud dorazily tyto zpravy: " *Zakladni otazka vseho* " od participanta **Student** , pokracuj

2.2. Vyvolej studenta

pokracuj

2.3. Vymysli odpoved

pokracuj

2.4. Odpovez studentovi

odesli zpravu " *42* " participantovi **Student**

pokracuj

2.5. Spokojene odejdi do kancelare vymyslet pekelnou pisemku

~~~~~~~~~~~~~~~~~~~~

Exportovani dokonceno

Obrázek B.2: vygenerované HTML v prohlžeči

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `pharo` ..... adresář s obrazem implementace
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF