ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

PROVOZNĚ EKONOMICKÁ FAKULTA

Katedra informačního inženýrství

Metody automatizované transformace modelů v analýze IS

Disertační práce z oboru Informační Management

Autor : Ing. Jakub Tůma

Školitel : doc. Ing. Vojtěch Merunka, Ph.D.

13. října 2016

Rád bych poděkoval svému školiteli doc. Ing. Vojtěchu Merunkovi, Ph.D. za odborné vedení po celou dobu mého doktorského studia a za rady při zpracování disertační práce. Dále bych rád poděkoval mým kolegům z Katedry informačního inženýrství. Děkuji své přítelkyni Veronice Vonešové za podporu, motivaci, lásku a za pomoc s formálními úpravami disertační práce. V neposlední řadě děkuji své rodině za všestrannou podporu.

Metody automatizované transformace modelů v analýze IS

Abstrakt

Tato doktorská disertační práce přispívá k holistickému vývoji informačních systémů v oblasti analytických modelů informačních systémů (IS). Práce se zabývá modely v analýze informačních systémů a jejich metody transformace.

Práce je zaměřena na konkrétní modely a to na Business process modeling notation (BPMN) a Business object relational modeling (BORM). Model BPMN je rozvíjen od roku 2000. Model BORM je starší a je rozvíjen od roku 1993. Obecným cílem této práce bylo rozšíření holistického vývoje informačních systémů. Konkrétním cílem bylo propojení modelu BPMN a modelu BORM.

Práce byla inspirována teorií konečných automatů. Stav problematiky řešení popisuje přístupy k transformacím modelů. V analytické části jsou matematickým zápisem popsány jednotlivé transformace, jakožto vstupní údaje pro realizační část. Realizační část obsahuje algoritmus transformace, postup jeho dosažení a jeho následné ověření na případových studiích. Diskuze obsahuje porovnání přístupu vytvořené metody transformace s ostatními přístupy.

Dosažení cílů je dokumentováno automatizovaným transformačním kalkulem. Přínosem je automatizované propojení modelů BPMN a modelu BORM pomocí metody transformace. Výsledkem je metoda automatizované transformace z modelu BPMN do modelu BORM pomocí algoritmu. Transformace modelu BPMN do modelu BORM je uskutečněna přes Mealyho automat.

Klíčová slova

Business Object Relation Modeling, Business Process Model and Notation, Algoritmus, Modelem řízená architektura, Transformace model na text, Transformace model na model, Konečný automat

Methods of Automated Model Transformations in Information System Analysis

Abstract

This doctoral dissertation thesis has impact to holistically development extension of information systems. Thesis has impact to analytical models and its transformation methods.

This thesis is focused on models Business Process Modeling Notation (BPMN) and Business Object Relational Modeling (BORM). Model BPMN is developed since 2000. Model BORM is developed since 1993 and it is older. The thesis general target was extension of holistically development of information systems. The main target was to bridge the gap between BPMN and BORM models.

This thesis was inspired by Finite State Machine (FSM). The state of the art describes approaches to models transformation. In the analytical part the mathematical formula are used to describe transformation. This part is used to base for the implementation part. The implementation part contains transformation algorithm its description and verification on the case studies. Impact of developed transformation method is validated on the comparison with the other approaches.

The targets were aimed and documented automated transformation calculus. The impact of automated BPMN and BORM models connection using transformation method. The result is method of automated transformation from BPMN model to BORM model using algorithm. The transformation method is based on the Mealy automaton.

Keywords

Business Object Relation Modeling, Business Process Model and Notation, Algorithm, Model Driven Architecture, Transformation Model to Model, Transformation Model to Text, Finite State Machine

Obsah

Seznam obrázků

Seznam tabulek

Matematické výrazy

Kapitola 1

Úvod

Tématem této disertační práce, jejíž název je Metody automatizované transformace modelů v analýze informačních systému, je oblast analytických modelů informačních systémů. Disertační práce je součástí skupiny celistvého (holistického) vývoje informačních systémů.

1.1 Cíl

Obecným cílem disertační práce je přispět k holistickému přístupu vývoje informačním systémům. Automatizace vývoje informačních systémů je postavena na možnosti automatickém vytvoření samotného informačního systému na základě požadavků.

Konkrétním cílem této disertační práce je spojení standardů BPMN a metody BORM pomocí techniky do takové míry podrobnosti, aby bylo možné tyto techniky implementovat matematickým zápisem, který bude popisovat transformace z modelu BPMN do modelu metody BORM.

Podcílem této disertační práce je analýza současného stavu v oblasti metody BORM, BPMN a transformací modelů, která je obsahem literární rešerše této práce. Obsahem literární rešerše je, jak lidsky čitelný zápis modelu, tak i strojově čitelný zápis modelu a přístupy k transformacím, které lze provádět s modely.

V této disertační práci budou vytvořeny metody, pomocí kterých se bude transfor-

movat analytický model podle jedné metody na model podle druhé metody.

Metoda se bude zabývat metodou BORM aplikovanou nad notací BPMN a jejich vzájemnou vazbou.

1.2 Hypotéza

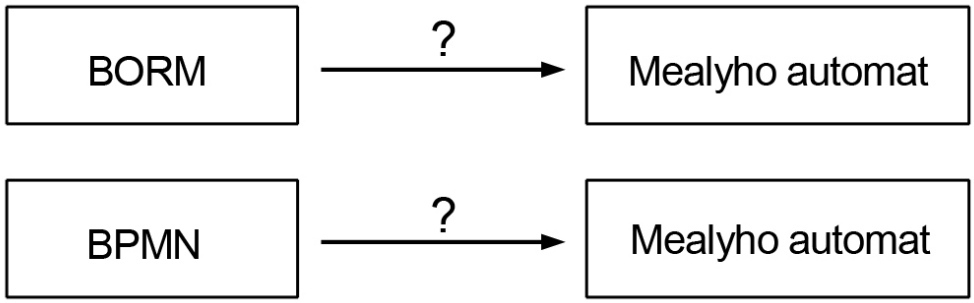
Úvodem do hypotézy, v souvislostech a nabraných vědomostech, lze vidět prostor v této oblasti a především přínos pro analýzu, návrh a implementaci systémů, které se vytvářejí v SOA (Service Oriented Architecture - servisně orientovaná architektura). Obchodní subjekty (např. obchodní analytici) používají a pracují se standardem pro modelování obchodních procesů BPMN (Business Process Model And Notation). Informační subjekty (např. informační analytici) používají a pracují se standardem pro modelování UML (Unified Modeling Language).Spojení například těchto dvou standardů by bylo možné pomocí transformačního nástroje, který by dokázal propojit BPMN a BORM. Fakt, že transformační nástroje mají význam, je podloženo vědeckými publikacemi od autorů (Koch, 2007) a (Mbarki & Erramdani, 2009). Tyto standardy (BPMN s UML) jsou součástí skupiny formálních specifikací Object Management Group (OMG), OMG® (OMG, 2011).

První hypotéza je formulována: Je možné propojit notaci BPMN a metodu BORM přes Mealyho automat. Základem první hypotézy je myšlenka, že společným bodem je Mealyho automat.

Druhá hypotéza je formulována: Je možné dosáhnout Mealyho automatu ze strany notace BPMN a metody BORM. Grafické znázornění druhé hypotézy, která vychází z první hypotézy, je vyobrazeno na obrázku 1.1 grafické znázornění druhé hypotézy na straně 3.

Hypotézy se opírají o závěry na základě vědeckých publikací (Koch, 2007), (Mbarki & Erramdani, 2009), (Montero et al., 2008).

Není stále dostačující nástroj, který bude podporovat transformace modelů (Koch, 2007). *"Dle našich znalostí představená metoda transformace není doposud známá."* (Montero et al., 2008). Tento fakt dává prostor pro vytvoření nové metody automatizované transformace modelů zaměřené na metodu BORM.



Obrázek 1.1: Grafické znázornění druhé hypotézy, autor

Lze argumentovat, že analytická část softwarového inženýrství je také založena na vytváření modelů, které jsou formálně specifikovány například pomocí skupiny OMG. Součástí skupiny OMG je široké spektrum formálních specifikací. Obchodními procesy se zabývá formální specifikace BPMN. UML formální specifikace se soustřeďuje na modelování informačních systémů z informačního hlediska.

1.3 Definice pojmů a terminologie

Pro potřeby této práce budeme softwarové inženýrství chápat v souladu se standardem IEEE 610.12:

Softwarové inženýrství je:

- *"Použití systematického, disciplinovaného, kvantifikovaného přístupu k vývoji, provozu a správě softwaru, tj. aplikací inženýrství do softwaru."*
- Studium přístupů jako v předchozím bodu.

Důležitými pojmy, které je třeba definovat pro potřeby disertační práce, jsou metodologie-metodika-metoda, které jsou přehledněji uvedeny v tabulce 1.1 Základní vědecké pojmy (Merunka, 2012*b*).

Tabulka 1.1: Vědecké pojmy metodologie-metodika-metoda od autora (Merunka, 2012*b*)

Pojem	Anglicky	Vysvětlení	Řecky
Metoda	Method	postup dosažení nějakého cíle (způsob cesty někam)	methodos
Metodika	nemá přímý ekvivalent v angličtině	pracovní postup nebo nauka o metodě	method -ikos = tvoří přídavná jména
Metodologie	Methodology	nauka o metodách, výklad metod	logos = slovo, projev, řeč
Technika	Technique	konkrétní nástroj nějaké metody či metodiky	techné = umění, řemeslo

Nebo také lze tyto tři pojmy chápat v souladu (Kadlec, 2004), tedy:

- *Metodologie* je nauka o metodách.
- *Metodika* je komplexní postup a návod (pro vývoj softwarové aplikace). Metodika zahrnuje více etap řešení (fází vývojového cyklu). Metodika odpovídá především na otázky typu proč?, kdo?, kdy?, co?, ale nemusí řešit otázky jak?
- *Metoda* je označení pro konkrétní postup vedoucí k vyřešení dílčího problému. Nezahrnuje více etap vývoje.

Pokud nebude řečeno jinak, samotný pojem metodika zahrnuje obě tyto varianty. Další termíny budou chápány v souladu s normou ČSN ISO 9000 (ČSN, 2006):

- *Proces* je soubor vzájemně souvisejících nebo vzájemně působících činností, který přeměňuje vstupy na výstupy.
- *Produkt* je výsledek procesu.
- *Postup* je specifikovaný způsob provádění činnosti nebo procesu.
- *Požadavek* je potřeba nebo očekávání, které jsou stanoveny, obecně se předpokládají, nebo jsou závazné.
- *Specifikovaný požadavek* je požadavek, který je stanoven například v dokumentu.

Kapitola 2

Metodický přístup

2.1 Teoretická část

Prvním krokem zpracování předkládané doktorské disertační práce byl v počátku průzkum stavu řešení a zpracování současného stavu problematiky. Jedná se o kapitolu obsahující literární rešerši ze zdrojů, které jsou uvedeny v seznamu literatury, a které jsou citovány podle vnitřních předpisů Provozně ekonomické fakulty České zemědělské univerzity v Praze. Čerpáno bylo především z publikací, monografií a tištěných či elektronických odborných periodik. Kapitola 3 Současný stav problematiky od strany 10 obsahuje literární rešerši zaměřenou na základní přístupy k modelování procesů a samotné modelování procesů.

První podkapitola literární rešerše 3.1 Úvod na straně 10 byla zaměřena na sběr teoretických poznatků o podnikových procesech, nástrojích pro modelování, teorii automatů a přístupech pracujících s UML. V rámci zpracování přehledu o současném stavu problematiky byly v úvodu kapitoly definovány principy modelování podnikových procesů, definován byl nástroj pro modelování procesů, popsány nutné teorie o automatech a také charakterizovány přístupy pracující s UML.

Podkapitola 3.1.5 Přístupy pracující s UML na straně 26 a podkapitola 3.1.6 RS-Lingo na straně 33 byly vypracovány v rámci doktorského zahraničního výjezdu podpořeného Českou zemědělskou univerzitou a programem Erasmus v roce 2015 do Portugalska. Tato část práce vznikla pod vedením externího školitele, kterým

byl Associate Professor[1] Alberto Manuel Rodrigues da Silva z Instituto de Engenharia de Sistemas e Computadores Investigacao e Desenvolvimento em Lisboa z Lisabonské technické univerzity.

Dále bylo v rámci kapitoly 3 Současný stav problematiky definováno a popsáno zobrazení procesů pomocí notací (BORM, BPMN), a to v podkapitole 3.2 Lidsky čitelný zápis modelu na straně 35. Ty byly vybrány na základě předběžných vědeckých prací předchůdců. Stěžejní notace metody BORM byly rozebrány detailněji a popsány byly jejich výhody. Samotná metoda BORM byla analyzována více podrobně, konkrétně její ORD (objektově relační diagram) byl popsán matematickým zápisem založeným na teorii automatů, které jsou definovány v rámci literární rešerše. V kapitole Lidsky čitelný zápis modelu jsou vstupní informace pro další výzkum. Reprezentace modelů ve strojovém zápisu byla rozebrána v podkapitole 3.3 Strojově čitelný zápis modelu na straně 63, která je rozdělena dále na podkapitoly Domain model, EMF a Standardy spojené s modely EMF.

V závěru kapitoly 3 Přehled současného stavu problematiky na straně 10 byly popsány přístupy transformací modelů a jejich kategorizace, a to v podkapitole 3.4 Transformace modelů na straně 65. Dále byl popsán přístup MDA (Model Driven Architecture) s návazností na kategorizaci transformací. Základním rozdělení transformace modelů bylo rozdělení na transformace model na model a transformace model na text. Kategorizace byla také provedena podle typu nástrojů. Podkapitola 3.4 Transformace modelů na straně 65 byla vypracována v rámci doktorského zahraničního výjezdu podpořeného Českou zemědělskou univerzitou a programem Erasmus v roce 2013 do Švýcarska na Universita della Svizzera italiana pod záštitou Associate Professor[2] Marc Langheinrich. Podkapitola 3.4.3 Transformace modelu na text na straně 67 a podkapitola 3.4.4 Transformace modelu na model na straně 69 obsahují informace, které vytváří rámec doktorského výzkumu.

2.2 Analytická část

Analytická část doktorské disertační práce se zabývá požadavky a návrhem řešení dané problematiky, které vychází ze zadání práce a z teoretických východisek vyplývajících z literární rešerše v kapitole 3 Současný stav problematiky na straně

[1] Associate Professor odpovídá titulu docent
[2] Associate Professor odpovídá titulu docent

10.

Předpoklady řešení → analýza řešení → návrh řešení

2.2.1 Požadavky řešení

Požadavky řešení v předkládané doktorské disertační práci byly rozděleny na dva na sebe logiky navazující kroky, a to na první krok předpoklady řešení a na druhý krok analýzu řešení. Předpoklady řešení byly formulovány v kapitole 1 Úvod na straně 1 společně s teoretickými hypotézami vyjádřenými v kapitole 1.2 Hypotézy na straně 2, které tato práce vědecky ověřuje. Druhý krok, analýza řešení, je formulován na základě sebraných informací z teoretické části výzkumu sepsaných v kapitole 3 Současný stav problematiky na straně 10 a popsán v kapitole 4 Analytická část na straně 75.

2.2.2 Návrh řešení

Studie transformace modelu na text → studie transformace modelu na model → popis metamodelů

Návrh řešení je opět rozdělen na jednotlivé na sebe logicky navazující kroky. Prvním krokem je studie transformace modelu na text v podkapitole 4.1. Metoda automatizované transformace modelu na text na straně 75, která byla provedena z modelu BORM ORD na textovou reprezentaci modelu. Druhým krokem, který vychází z provedené studie transformace modelu na text, je návrh samotné transformace modelu na model, popsaný v podkapitole 4.2 Metoda automatizované transformace modelu na model (modleu BPMN PSD na model BORM ORD) na straně 78.

Součástí návrhu řešení jsou metamodely obou diagramů notací - jak zdrojového, tak také cílového, respktive BPMN PSD a BORM ORD modelu.

Dalším krokem v návrhu řešení je popis metamodelů obou notací - jak zdrojového, tak také cílového, respektive BPMN PSD a BORM ORD modelu. Nalezeny a formulovány jsou spojitosti jednotlivých elementů v metamodelech zdrojového a cílového modelu. Formulovány byly definice spojitostí pomocí tabulky a matematického vyjádření, které bylo založeno na teorii automatů a Petriho síti.

2.3 Realizační část

2.3.1 Implemetace

Implementace automatizované metody transformace modelu BPMN PSD na model BORM ORD byla provedena pomocí algoritmu popsaného v podkapitole 5.2.1 Transformace BPMN PSD na BORM ORD na straně 101. Tento algoritmus byl formulován na základě návrhu řešení. Ladění a ověřování algoritmu bylo prováděno na datech, které byly získány z případových studií, které následně sloužili také k verifikaci a validaci.

2.3.2 Verifikace a validace

Provádění a ověřování algoritmu bylo realizováno na případových studiích, které jsou popsány v podkapitole 5.3 Případové studie na straně 103. Data pro tyto případové studie byly sebrány v rámci předmětu Informační management vyučovaného na Provozně ekonomické fakultě České zemědělské univerzity v Praze, a to na cvičeních, které jsem vedl. Pro splnění podmínek udělení zápočtu sbírali studenti data pro jednotlivé případové studie, které byly posléze zpracovány autorem, jakožto data sloužící pro ověření výsledků této předkládané doktorské disertační práce. Případová studie uvedené v této práci jsou pouze ilustrací zpracovaných studií. Kompletní informace o jednotlivých případových studiích využitých v rámci tohoto výzkumu, jsou uvedeny v Přílohách na straně 143. Konkrétně se jedná o tyto případové studie:

1. Agendy zahraničního oddělení fakulty vysoké školy popsaná v kapitole 5.3.1 na straně 104.
2. Elektronický obchod popsaná v kapitole 5.3.2 na straně 105.
3. Metoda pachové identifikace popsaná v kapitole 5.3.3 na straně 106.

Ověření opodstatnění výsledků této disertační práce bylo provedeno diskuzí, která je popsána v kapitole 6 Shrnutí výsledků a diskuze na straně 108.

Validace

Ověření algoritmu bylo prováděno automatizovaně na základě zpracovaných jednotlivých modelů následujícím způsobem:

1. Nejprve byl vytvořen model ve výchozí notaci BPMN, který byl zakreslen v nástroji pro modelování procesů BPMN. V tomto konkrétním případě tohoto výzkumu byl využit nástroj yaoqiang-bpmn-editor.
2. Takovýto vzniklý model byl převeden a následně zakódován na jednotlivé elementy a jejich vlastnosti co byly definovány dle metamodelu 10.21 Metamodel PSDiagram BPMN uvedeného v Přílohách na straně 155. Převod a kódování bylo provedeno podle tabulky 5.2 Tabulka zobrazení a transformační pravidla na straně 100 a jednotlivé elementy byly očíslovány pro transformaci.
3. Nad takto zakódovanými elementy byla provedena transformace podle popsaného algoritmu 5.1 Transformace BPMN PSD na BORM ORD na straně 101.
4. Následně bylo provedeno zpětné odkódování elementů a zakreslení výsledného modelu do notace metody BORM v nástrooji OpenCase.
5. Poté byla provedena optimalizace výsledného modelu metody BORM.

Pro ověření algoritmu byly jako příklad vstupu a výstupu tohoto postupu použity zmiňované ilustrativní případové studie popsané v kapitole 5.3 Případové studie na straně 103 a uvedené v Přílohách na straně 143.

Kapitola 3

Současný stav problematiky

3.1 Úvod

Obor, zabývající se procesy je znám po dlouhou dobu, ale během poslední dvacítky let je více atraktivní (Lindsay et al., 2003). Toto je z důvodu pokroku v informačních technologiích, a také protože informační technologie hrají větší roli v organizačním řízení. V současnosti je nejznámějším oborem řízení podnikových procesů (BPM - Business Process Management) .

Řízení podnikových procesů je rámec zaměřený na řízení, který oproti ostatním technikám obsahuje modelování podnikových procesů a simulaci podnikových procesů. Modelování podnikových procesů je používáno za různými účely například dokumentace procesů, simulace procesů a optimalizace, dále specifikace požadavků, softwarové inženýrství a implementace procesně orientovaných informačních systémů.

Počet dostupných metod a metodologií zabývající se procesním modelováním je velmi rozsáhlý a stále roste. Přehled obsahuje 25 metodologií, 72 technik a 102 nástrojů pro procesní modelování. Procesy mohou být modelovány z různých perspektiv, každá metodologie byla vyvinuta pro poskytnutí odlišného účelu, a tím přináší své výhody (Kettinger et al., 1997) a (Aguilar-Saven, 2004). Tato rešerše bude zaměřena na BORM (Business Object Relationship Modeling) a BPMN (Business Process Model and Notation).

BORM metodologie je objektově orientovaná a procesně založená metodologie pro

analýzu a návrh, která se ve své první fázi zaměřuje na zachycení procesu v organizaci (Polák et al., 2003). Cílem této metodologie je překlenout mezeru mezi podnikovým inženýrstvím (business engineering) a softwarovým inženýrstvím (software engineering) .

BPMN je celosvětově známý standard, který definuje notaci pro podnikové procesní modely. Má široké spektrum použití a zachycuje provozní detaily procesu, detailně popsané ve specifikaci od skupiny OMG (OMG, 2006).

3.1.1 Modelování podnikových procesů

Modelování podnikových procesů je přístupem k popisu, jak organizace řídí současné nebo budoucí podnikové procesy. (Indulska et al., 2009) modelování podnikových procesů je obrazová reprezentace procesu. Zpravidla používá některý modelovací přístup pro dosažení konce použití modelování. Modelování zahrnuje shromažďování informací o samotném procesu a z okolí podnikového procesu, takže proces může být optimalizován (Ganesan, 2010).

Mapování [1] podnikových procesů a modelování podnikových procesů nejsou stejné termíny (Indulska et al., 2009). Mapování podnikových procesů znamená představení konečných detailů a může, nebo nemusí být použito k analýze procesu. Mapování podnikových procesů je použito pro konkrétní účely procesní reprezentace, například při komunikaci co se děje v samotném procesu. Modelování podnikových procesů zachycuje data, role, zdroje a další detaily, které jsou potřebné pro simulaci procesu. (Ganesan, 2010) V souvislosti s modelováním podnikových procesů jsou používány pojmy model, model procesu nebo konceptuální model procesu. Model je popis, stejně jako abstrakce systému (Banks et al., 1998). Konceptuální model představuje koncepty (entity) a vztahy mezi nimi, podle autora (Hommes, 2004). Model procesu je konceptem podnikového procesu v praxi (enterprise) (Dietz, 2006).

Modely procesů popisují, typicky v grafické úpravě, aktivity, události a logický řídící tok, který představuje podnikový proces (Recker et al., 2009).

Modely procesů jsou považovány za klíčové nástroje pro analýzu a návrh procesně založených informačních systémů, dokumentace organizací a její re-inženýrství a návrh servisně orientované architektury (Indulska et al., 2009). Modelování procesů

[1] Slovo mapování znamená v tomto smyslu zobrazení (přeloženo z anglického slova mapping).

je uskutečněno z mnoha důvodů (Ko et al., 2009). Tři z nich:

- Prvním důvodem je podnikové modelování.
- Druhým důvodem je modelování podnikových procesů.
- Třetí důvod je vývoj informačních systémů.

Podnikové modelování je předběžným stupeněm k podnikovému inženýrství. Podnikové inženýrství pracuje s analýzou, návrhem, inženýrstvím a implementací v podniku (Hommes, 2004).

Modelování podnikových procesů hraje kritickou roli v úspěšném inženýrství složitých systémů, protože za první krok je považováno porozumění procesům v organizaci (Hommes, 2004). Úspěšná systémová implementace začíná s porozuměním podnikových procesů v organizaci (Aguilar-Saven, 2004).

Podle perspektivy řízení organizace, modelování podnikových procesů je výhodné, jak ve strategickém, tak i v provozním řízení. Strategické cíle, jako spokojenost zákazníka a zisk, vysvětlují, proč by měl být určitý proces řízen určitým směrem. Analýza cílů strategických výsledků je v definici procesů, zatímco provozní cíle se starají o jednotlivé procesy a jejich operace (Bider, 2005). Podle přehledu spočívá nejvíce výhod je v oblasti organizační, manažerské a provozní. Modelování procesů má výhodu ve zdokonalení procesu, porozumění a komunikaci (Indulska et al., 2009).

Modelování obchodních[2] procesů je vždy řízeno s tím, že je brán v potaz předpokládaný účel. Aby byla zvolena správná technika, tvůrce modelu musí znát účel konstruovaného modelu, protože různé techniky jsou shodné pro odlišné účely (Aguilar-Saven, 2004).

Framework pro klasifikaci modelovacích technik obchodních procesů ve kterém se rozlišují čtyři účely modelů obchodních procesů (Aguilar-Saven, 2004):

- Prvním účelem jsou popisné modely pro výuku.
- Druhým účelem jsou popisné a analytické modely pro rozhodovací podporu vývoje a návrhu procesů.

[2]Slovo obchodních je přeloženo z anglického slova business, které je obecnější než jen obchod, ale znamená také administrativa, úřad a další.

- Třetím účelem jsou analytické modely pro podporu rozhodování během výkonu a řízení procesu.
- Čtvrtým účelem je podpora předpisu modelů pro informační technologie.

Definuje se odlišné rozdělení podle účelu procesního modelování, které je založeno na jeho použití (Phalp, 1998):

- Zachycení je použito pro vývoj softwaru. Toto je zaměřeno na zachycení čitelného a srozumitelného pohledu na obchodní proces. Uživatelé raději pozorují model, než s ním interagují.
- Analyzování je použito pro modely. Ty by měly odpovídat dynamickým a funkčním vlastnostem procesu. Obvykle uživatelé chtějí pracovat s procesem, například používat simulaci, aby byli schopni odpovědět na otázku, co se stane když?
- Prezentace je použita pro modely. Ty by měly být snadné k porozumění, když jsou používány k dokumentaci procesů. Typické je použití notace diagramů, které je schůdné.

Rozdělení pomocí použití v oblastech modelování procesů je na následující (Krogstie, 2012):

- V lidském vnímání zahrnuje popis současného stavu (AS-IS modelování).
- Komunikace je mezi lidmi v organizaci.
- Počítačově asistovaná analýza je použita k získání znalostí o organizaci přes simulaci a porovnání stavu AS-IS a TO-BE stavu.
- Zaručení kvality zajišťuje organizaci, která pracuje podle certifikovaného procesu.
- Modely nasazení a aktivace mohou být aktivovány přes lidi nebo automaticky za použití workflow[3] systémů.
- Je vstupem do tradičních vývojových projektů.

[3]Anglické slovo workflow představuje v tomto spojení tok nebo také posloupnost práce.

Procesy organizací jsou běžně velmi komplexní a shodují se složitostí celkových konceptuálních modelů. Je běžné, že se model rozdělí na několik dalších modelů. Obecně se modely dělí na statické a dymanamické typy modelů. Základní perspektivy jsou tři, a to data, procesy a chování. Datové perspektivy jsou zaměřeny na statické vlastnosti, procesy a chování je zaměřeno na dynamické vlastnosti (Hommes, 2004). Obecně znamená perspektiva úhel pohledu, který je zaměřen na vytváření modelu procesu.

V konceptuálním modelování je technika tělem technické znalosti, která vede toho, kdo modeluje, v průběhu konstruování skutečnosti do konceptuálního modelu (Hommes, 2004).

Při modelování procesů je hlavním zaměrem zachycení dynamiky procesu. Dynamika procesu může být modelována čtyřmi způsoby (Bider, 2005):

- Vstupně výstupní toky se zaměřují na pasivní účastníky, kteří jsou pomocí aktivit produkování, spotřebováni nebo změny. Nejběžnější technikou pro tuto perspektivu je IDEF0.
- Workflow je zaměřeno na pořadí aktivit v čase. Typické workflow diagramy jsou : IDEF3 , Action diagram z UML a Petriho sítě .
- Stavový tok je reprezentován tak, že v každé aktivitě je produkována změna v části ze skutečného světa. Tok je popsán v diagramu přechodů.
- Pohled orientovaný na agenty je zaměřen na pořadí, ve kterém agenti dostávají a vykonávají svoji práci. Typická notace je *diagram aktivit*, rolí a kolaborační diagramy v UML.

Metoda by neměla být zaměňována s technikou. Metoda se týká uspořádaného pořadí způsobu práce. Metoda obsahuje vzájemně propojené úkony, které musí být splněny, aby se dosáhlo požadovaného cíle, jak uvádí autor (Hommes, 2004).

Metodologie je tělo metod, pravidel a předpokladů. Definuje slovo metodologie jako částečnou proceduru nebo množinu procedur (www.merriam webster.com, 2014).

3.1.2 Vybrané nástroje pro modelování procesů

Moravec (2014) uvádí ve své práci, že návrh informačních systémů usnadňují CASE nástroje. Tyto nástroje samy o sobě nestačí k vytvoření informačního systému podle požadavků uživatele, což uvádí autoři Shlaer & Mellor (1992) ve své publikaci. Nicméně, ani dokonalé znalosti a zvládnutí práce s CASE nástrojem nezaručují dobrý výsledek. Pro vytvoření informačního systému je důležité pochopit problém a procesy uživatele, které má informační systém pokrývat. Tedy dokonalé zvládnutí CASE nástroje nepostačuje k pokrytí dobrého výsledku. Zásadní pro vytvoření informačního systému je pochopení, co je pro uživatele podstatné, jak uvádějí autoři Knott et al. (2000), Polák et al. (2003) ve svých publikacích. Dále je nutné překlenout mezeru mezi uživateli a realizátory informačního systému.

Překlenutí této mezery je úkolem architektů informačních systémů (da Silva, 2003). Transformaci uživatelského úhlu pohledu na zadání pro realizaci metody řeší architekti informačních systémů (da Silva, 2003).

Spojnice mezi realizací a architektem informačního systému je systémový integrátor. Z mnoha definic systémového integrátora je jedna uvedená v publikaci od kolektivu autorů Habáň & Sodomka (2004): *"Systémový integrátor je subjekt, který má přímý vztah k zákazníkům a je vždy spoluřešitelem IS/IT projektů u zákaznických organizací."*

Jiná definice je podle kolektivu autorů Polák et al. (2003): Systémový integrátor je subjekt, který učelně umí nalézt a pospojovat komponenty informačního systému s cílem naplnění požadavků a procesů uživatele informačního systému.

Historický vývoj Business Process Management (BPM) započínal v osmdesátých letech dvacátém století , kdy se začal používat TQM (Total Quality Management). Filozofie řízení TQM má za cíl trvalé zvyšování kvality procesů a produktů. TQM je předchůdce BPR (Business Process Reingeneering) , která se začlo rozšiřovat v devadesátých letech dvacátého století. Cílem BPR je zvýšení efektivity procesů v organizaci pomocí analýzy, návrhu a realizace v podobě přepracování jich samých, jak uvádějí autoři Jeston & Nelis (2014).

Návrh, analýza a řízení obchodních procesů je obsahem Business Process Management (BPM). Využívá k tomu metody, nástroje a techniky. Tento směr navazuje na předchozí BPR a objevuje se v první dekádě dvacátého prvního století, což

publikoval kolektiv autorů Ouvans et al. (2006).

Cílem BPM oproti BPR je i procesy měřit. Jinak má BPM za cíl, jako u BPR, řízení procesů a zvyšování jejich efektivity. Obecnější definice podle autorů Jeston & Nelis (2014) říká, že obsahem BPM jsou obchodní procesy. Přístup BPM je interdisciplinární a zahrnuje management a informační technologie.

3.1.3 Podnikový proces

Obchodní proces , nebo také podnikový proces, což je ekvivalentní s obchodním případem, definuje autor Ould (2005). Definice podnikového procesu podle autora Ould (2005): *"Podnikový proces je složen z množiny aktivit navzájem propojených, které jsou prováděny účastníky. Účastníci procesu navzájem kooperují. Smyslem podnikového procesu je dosažení konkrétního cíle."*

Podobným postupem definují obchodní proces ve své publikaci autoři Smith & Fingar (2003): Ti definují obchodní proces jako množinu, která je celistvá a dynamicky koordinovaná. Množina obsahuje aktivity, které spolupracují a jsou transakčně provázané. Cílem aktivit je přinášet hodnotu pro zákazníka.

Obchodní proces transformuje informace vstupní na informaci výstupní. Výsledkem obchodního procesu je pro zákazníka přidaná hodnota. Syntézou definic obchodního procesu vznikne výsledek s posloupností stavů a aktivit, které jsou vzájemně provázány a mají konečný a výchozí stav.

Autoři Jeston & Nelis (2014) popisují ve své publikaci vlastnosti modelů obchodních procesů. Formální základ u procesních modelů dovoluje úplný a jednoznačný popis a tím i lepší analýzu. Přehledné grafické vyjádření procesu by mělo usnadnit porozumění pro každou zainteresovanou stranu. Jednoznačnost výkladu by měla být srozumitelná pro každého čtenáře procesního modelu (vlastníci, účastníci, vývojáři i analytici), tak aby nebyla umožňěna jiná interpretace.

Přístupy modelování podnikových procesů

Podnikový případ je obdobný název pro obchodní proces, který je překladem z anglického názvu Business Process. Obchodní proces je možné označit jako obchodní

případ.

Cílem modelování procesů je ukázat místa s rozhodováním v procesu a klíčová místa.

Přes širokou škálu využívaných přístupů a technik modelování podnikových procesů budou představeni zástupci dle přehledu od autora Giaglis (2001).

Flowcharting

Flowcharting jsou označovány jako vývojové diagramy. Historie vývojových diagramů se datuje do šedesátých let dvacátého století. V počátku byly vývojové diagramy užívány pro zachycení logiky softwaru a abstrakce algoritmů. Vývojové diagramy jsou velice obecným přístupem, je tedy možné je použít i pro modelování podnikových procesů. Tento přístup je používán pouze okrajově pro modelování podnikových procesů. Výhoda vývojových diagramů je komplexnost zobrazení toků informací a skladby procesu.

Teorie grafů

Definice v oblasti teorie grafů byly čerpány z publikací od autorů Bondy & Murty (2008) a Hliněný (2007). Definice se řídí následující konvencí. [4]

Definice 3.1. Graf G je uspořádaný pár $\langle V_G, E_G \rangle$ obsahující množinu V_G uzlů a množinu hran E_G, která je disjunktní od V_G (Bondy & Murty, 2008).

Definice 3.2. Procházka v grafu **G** je konečná nenulová sekvence $W = \langle v_0, e_0, v_1, e_1, v_2, e_2, ..., v_k e_k \rangle$, jejíž termy jsou střídavě uzly a hrany, pro které platí $1 \leq i \leq k$, konec e_i jsou v_{i-1} a v_i .

Definice 3.3. Lze říci, že **W** je procházka z v_0 do v_k, nebo také (v_0, v_k) procházka. Uzly v_0 a v_k jsou nazývány počátek a konec W, respektive jeho vnitřní uzly jsou $\langle v_1, v_2, v_3, ..., v_{k-1} \rangle$. Celé číslo **k** je délka procházky (Bondy & Murty, 2008).

[4] V definici se používá konvence $G = \langle V_G, E_G \rangle$ jako $V_G \cup E_G = \emptyset$, kde $\langle a, b, c, ... \rangle$ uspořádaná n-tice a (a, b, c, ...) je výčet - neuspořádaná n-tice

Definice 3.4. Cesta je název pro procházku, pokud hrany $(e_1, e_2, e_3..., e_k)$ jsou odlišné a současně jsou v procházce odlišné uzly $(v_1, v_2, v_3, ..., v_k)$ (Bondy & Murty, 2008).

Definice 3.5. Spojitý graf je takový graf **G**, který má právě jednu **komponentu,** jinak se jedná o nespojitý graf. Dva uzly **u** a **v** grafu **G** jsou propojené, pokud existuje **cesta (u,v).** Propojení je ekvivalentní vztah na množině uzlů **V.** Pokud existuje rozdělení **V** na neprázdné podmnožiny $(V_1, V_2, V_3..., V_w)$, kde dva uzly **u** a **v** patří do stejné množiny V_i, potom podgrafy $(G_{V_1}, G_{V_2}, G_{V_3}, ..., G_{V_w})$ se jmenují **komponenty** grafu **G** (Bondy & Murty, 2008).

Definice 3.6. Cyklus v grafu **G** je pokud se v něm nachází cesta a počáteční a koncový uzel jsou totožné. Délka cyklu se značí **k-cyklus** , kde **k** značí jeho délku a určuje, zda je cyklus lichý nebo sudý (Bondy & Murty, 2008).

Definice 3.7. Bipartitní graf je takový graf, který neobsahuje liché cykly (Bondy & Murty, 2008).

Definice 3.8. Orientovaný graf je uspořádaná dvojice $G = \langle V, E \rangle$, kde $E \subseteq V \times V$. Hrana $E = \langle u, v \rangle$ začíná ve vrcholu (uzlu) **u** a končí (míří do) v uzlu **v** (Hliněný, 2007).

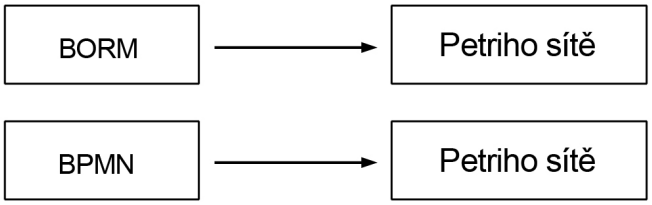
Definice 3.9. Cyklický graf je graf, který má alespoň jeden cyklus (Hliněný, 2007).

Definice 3.10. Acyklický graf je graf, který není cyklický, tedy neobsahuje žádný cyklus (Hliněný, 2007).

Definice 3.11. Orientovaný acyklický graf je orientovaný graf, kde není možné nalézt orientované cykly (Bondy & Murty, 2008).

Petriho sítě

Pro modelování paralelních výpočtů a výrobních procesů jsou velice často využívány Petriho sitě. Petriho sítě je možné využít i mimo technickou sféru, například v obchodních procesech, ve kterých je lze využít k analýze, modelování, popisu a simulaci, jak uvádí autoři Girault & Valk (2003).



Obrázek 3.1: BORM a BPMN převod na Petriho sítě, autor

Z pohledu teorie grafů je Petriho síť ohodnocený, orientovaný, bipartitní graf. Tato práci se zaměřuje pouze na základní Petriho sítě, které se skládají z hran, míst a přechodů. Oblastí zkoumání Petriho sítí jsou procesy, které jsou reprezentované matematicko-graficky. Paralelní systémy a jejich dynamika a struktura chování jsou analyzovány pomocí Petriho sítí.

Petriho síť je trojice

$$N = \langle P, T, F \rangle, \tag{3.12}$$

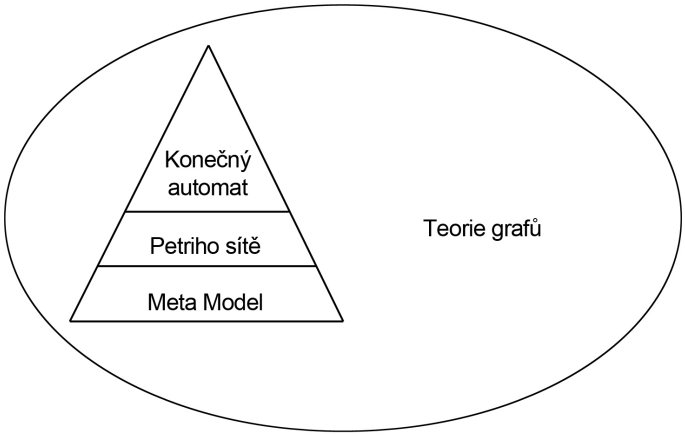
kde:
P a T jsou disjunktní konečné množiny míst a přechodů.

$$F \subset (P \times T) \cup (T \times P) \tag{3.13}$$

F je množina hran (sekvenční tok).

Petriho sítě jsou používané v oblasti obchodních procesů, jak již bylo zmíněno. Pomocí Petriho sítí jsou popsány syntaxe a sémantika notace BPMN (Dijkman & Dumas, 2007). Na druhé straně je také popsána Petriho sítěmi metoda BORM, kterou publikoval Papík (2005). Tento postup je znázorněn na obrázku 3.1 na straně 19.

Podle definic jednotlivých pojmů v této práci je hlavní množinou je teorie grafů, ve které jsou obsaženy konečné automaty, Petriho sítě a metamodely. Hierarchické uspořádání konečných automatů, Petriho sítí a metamodelu je znázorněno pyramidou, kde konečné automaty jsou na vrcholu a Petriho sítě spolu s metamodelem



Obrázek 3.2: Grafické znázornění metody transformace BORM na BPMN, autor

jsou znázorněny pod konečnými automaty. Celé toto uspořádání je zobrazeno na obrázku 3.2 na straně 20.

3.1.4 Teorie automatů

Moravec (2014) píše:"*Historie teorie automatů se datuje k roku 1943, kdy navrhli McCuloch a Pittsov model zařízení. Tento model zařízení je abstraktní a má konečný počet dosažitelných stavů. Stavové automaty jsou způsob jak modelovat chování systémů.*"Tento způsob je velice známý a starý, jak uvádějí autoři Gill (1962) a Wright (2005).

Proces lze popsat pomocí konenčného automatu v konkrétním místě a čase, stejně tak jako stav systému. Na základě stavů je možné odvodit chování celého systému. Tato technika má výhodu, že je nezávislá na implementaci a také na volbě konkrétního nástroje.

Konečný automat

Determinictický konenčný automat (Shallit & Shallit, 2009) je nejjednodušší model počítače. Formálně deterministický konečný automat je šestice $\langle S, A, B, \psi, s_0, S_f \rangle$, kde

- S je konečná neprázdná množina stavů

- A je konečná neprázdná množina vstupní abecedy
- B je konečná neprázdná množina výstupní abecedy
- $\psi : S \times A \to S$ je přechodová funkce
- $s_0 \in S$ je start nebo počáteční stav
- $S_f \subset S$ je množina konenčých stavů

Mealyho automat pro množiny A a B, kde

$$(S, \psi) \tag{3.14}$$

s vstupy A a výstupy B obsahuje stavy S a přechodovou funkci $\psi : S \to (B \times S)A$. Tato funkce propojuje stav $s0 \in S$ na funkci $\psi(s0) : A \to (B \times S)$, která produkuje pro každý vstup $a \in A$ pár (b, s1), kde b je výstup a s1 je následující stav. $\mu : S \times A \to B$ je výstupní funkce. $s_0 \in S$ je počáteční stav, a $s_f \in S$ je koncový stav.

$$\begin{aligned} \psi : S \to (B \times S)A \\ s0 \in S \psi(s0) : A \to (B \times S) a \in A \\ \mu : S \times A \to B \\ s_0 \in S \\ s_f \in S \end{aligned} \tag{3.15}$$

Tedy Mealyho automat je $\langle S, A, B, \psi, \mu, s_0, s_f \rangle$ (Shallit & Shallit, 2009), (West, 2002) a (Mealy, 1955).

Moorův automat Definice Moorova automatu (Moore, 1956) je obdobná definici Mealyho automatu s tím rozdílem, že výstupní funkce je odlišná v tomto:

$$\mu : S \to B \tag{3.16}$$

Transformace Moorova automatu na Mealyho automat

Definice 3.17. Puntambekar (2008) ve své knize píše, že metoda transformace Moorova automatu na Mealyho automat je následující: mějme pětici M reprezentující Moorův automat a pětici $M^{'}$, která reprezentuje Mealyho automat. Potom výstupní funkce $\mu^{'}(s,a) = \mu(\psi(s,a))$. Kde $M^{'}$ a M jsou definovány jako v kapitole 3.1.4 Mealyho automat na straně 21.

$$\begin{aligned} M^{'} &= \langle S, A, \psi, \mu^{'}, s_0, s_f \rangle \\ M &= \langle S, A, \psi, \mu, s_0, s_f \rangle \end{aligned} \tag{3.18}$$

Tato transformace je také popsána například v článku autora Babcsanyi (2000). A oproti tomu autoři knihy Shallit & Shallit (2009) píší i o transformaci z Moorova automatu na Mealyho automat a také u transformace Mealyho automatu na Moorův automat uvádí důkazy.

Pro vstup $x = a_1a_2...a_n$ Moorův automat M přechází do stavů
$s_0, \psi(s_0,a_1), \psi(s_0,a_1a_2), ..., \psi(s_0,a_1a_2...a_n)$ a výstupů
$y = \mu(s_0)\mu(\psi(s_0,a_1))\mu(\psi(s_0,a_1a_2))...\mu(\psi(s_0,a_1a_2...a_n))$. Tedy lze psát, že $T_M(x) = y$. Mealyho automat má výstup $z = \mu(s_0,a_1)\mu(\psi(s_0,a_1),a_2)...\mu(\psi(s_0,a_1a_2...a_{n-1}),a_n)$.

Píše se tedy $U_M(x) = z$. Lze si všimnout, že pro vstup délky n má Moorův automat výstup délky $n+1$ a Mealyho automat má výstup délky n. Tento fakt nastává, jelikož Moorův automat vždy poskytuje výstup spojený s počátečním stavem. V některém smyslu tento výstup není podstatný, neboť nezáleží na vstupu.

I přes tento rozdíl lze definovat ekvivalenci mezi automatem typu Moore a automatem typu Mealy. Moorův automat M je ekvivalentní Mealyho automatu $M^{'}$, pokud jejcih vstupní a výstupní chování je stejné, vyjma prvního výstupu Moorova automatu. Formálně lze psát M je ekvivalentní $M^{'}$, když pro včechny $x \in \sum^*$, je $T_M(x) = T_M(\epsilon)U_{M'}(x)$.

Věta 3.18.1. Nechť $M = \langle S, A, B, \psi, \mu, s_0 \rangle$ je Moorův automat. Potom existuje ekvivalentní Mealyho automat $M^{'}$ se stejným počtem stavů.

Myšlenka důkazu je definování výstupní funkce $\mu^{'}$ simulujícího automatu tak, že

jeho hodnota záleží na výstupu ze stavu, který je dosažen, potom co je vykonán přechod.

Důkaz transformace Moorova automatu na Mealyho automat. Formálně pro $M^{'} = \langle S, A, B, \psi, \mu^{'}, s_0 \rangle$, kde $\mu^{'}(s, b) = \mu(\psi(s, b))$. Když vstup pro M je $x = a_1 a_2 ... a_n$, potom $M^{'}$ má výstup

$$\mu^{'}(s_0, a_1)\mu^{'}(\psi(s_0, a_1), a_2)\mu^{'}(\psi(s_0, a_1 a_2), a_3)...\mu^{'}(\psi(s_0, a_1 a_2 ... a_{n-1}), a_n). \tag{3.19}$$

Podle definice $\mu^{'}$ je

$$\begin{aligned} &\mu(\psi(s_0, a_1))\mu(\psi(\psi(s_0, a_1), a_2))...\mu(\psi(\psi(s_0, a_1 a_2 ... a_{n-1}), a_n)) \\ &\quad = \mu(\psi(s_0, a_1))\mu(\psi(s_0, a_1 a_2))...\mu(\psi(s_0, a_1 a_2 ... a_n)). \end{aligned} \tag{3.20}$$

Toto následuje $T_M(x) = T_M(\epsilon)U_{M'}(x)$. ■

Komunikující konečné automaty Moravec (2014) píše: "*Komunikující konečný automat (CFSM – Communicating Finite-State Machine) se používá pro komunikační protokoly.*"Jde o abstraktní model, který se využívá pro návrh, testování, syntézu, validaci a specifikaci těchto protokolů (Brand & Zafiropulo, 1983). Konečné automaty jsou využity pro popis procesů. Procesy mezi sebou komunikují v podobě konečných automatů přes zásobník. Teoretický zásobník je neomezeně velký a pomocí něj zasílány a přijímány asynchronně zprávy.

Prostředkem pro komunikaci je zásobník, do kterého se ukládají zprávy, které nebyly přijaty, ale byly odeslány. Chyby jsou hlavním problémem. Nesmí dojít k zastavení přechodů mezi stavy automatu z důvodu chyb v komunikaci.

K přerušení komunikace dochází ze tří příčin. Dojde-li k uváznutí procesu, nevymezené komunikaci a nespecifikované přijímání zpráv, tak se komunikace přeruší (Brand & Zafiropulo, 1983).

Na výzkum Brand & Zafiropulo (1983) navazují Peng & Puroshothaman (1991),

jejichž přístup umožňuje zabránit nespecifikovanému přijímání zpráv a uváznutí procesu. Toto je možné omezit pomocí analýzy jednotlivých automatů a jejich datových toků. Podmínky pro komunikaci jsou nastaveny tak, aby jeden konečný automat vyjadřoval pouze jeden proces, a mezi těmito konečnými automaty byly použity zásobníky jako komunikační kanály výhradně pro komunikaci.

Komunikační kanál je realizován plně duplexním kanálem typu FIFO, který propojuje právě dva procesy. Autoři Peng & Puroshothaman (1991) navazují na práci Brand & Zafiropulo (1983) a jejich práce oproti Brandovi a Zafiropulovi (1983) obsahuje modelování procesů. Zatímco práce Branda a Zafiropula (1983) obsahuje aplikaci návrhu komunikačních protokolů. Autoři Peng a Purosthaman (1991) popisují ve své publikaci CFSM vzájemnou komunikaci dvou procesů a možnost nastavení stavů a komunikačních podmínek. Komunikující konečné automaty (CFSM) jsou implementované dle návrhu autorů Penga a Purosthama (1991).

Když jsou využity komunikující konečné automaty, je možné popsat nejen průběh dvou procesů, ale také uceleného seskupení procesů, jak píší autoři Peng & Puroshothaman (1991). Množina stavů a událostí procesního modelu je vždy konečná, protože je uvažována analytikem nebo návrhářem. Procesní model může být reprezentován seskupením konečných automatů (CFSM), nebo například různým procesním diagramem. Seskupení konečných automatů lze zobrazit na orientovaný graf, u kterého je definovaný počáteční stav, kde stav odpovídá uzlu. Zobrazení hrany je v orientovaném grafu na právě jednu aktivitu a aktivita odpovídá události. Přijímané a odesílané zprávy jsou z konečné množiny všech zpráv a jsou odesílány a přijímány z aktivit. Tento model je aplikován v metodě BORM. Jak píší autoři Peng & Puroshothaman (1991) komunikace mezi CFSM je asynchronní.

Vzhledem k tomu, že komunikace je asynchronní, je nutné použít nekonečně velký zásobník typu FIFO. Zásobník typu FIFO se používá pro ukládání čekajících zpráv na zpracování. Tedy mezi každýma dvěma vzájemně propojenýma komunikujícíma CFSM je potřeba mít vložen tento zásobník. Oproti tomuto typu komunikace je popsán typ komunikace s nulovým časem prodlevy mezi odesláním a přijetím zprávy mezi komunikujícími automaty (Gouda et al., 1984). Zpráva je ve stejném čase odeslána i přijata a zpracována. Zavedení tohoto typu komunikace CFSM neobsahuje asynchronní komunikaci a může docházet k uváznutí a k dosažení nespustitelných přechodů.

Definice 3.21. Definice CFSM podle Peng & Puroshothaman (1991) je P_i jako

uspořádanou čtveřici:

$$\langle S_i, \langle \sum_{i,j} \rangle_{j \in I} \cup \langle \sum_{i,j} \rangle_{j \in I}, \delta_i, s_{0i} \rangle, \text{ kde:} \tag{3.22}$$

- S_i je konečná neprázdná množina stavů automatu i.
- I je množina všech komunikujících konečných automatů tvořících síť, I = 1, . . . , n, kde $n \geq 2$.
- $\sum_{i,j}$ je množina druhů zpráv, které může automat P_i odeslat automatu P_j a $\sum_{j,i}$ je množina druhů zpráv, které může automat P_i přijmout od automatu P_j. Předpokládáme, že $\sum_{i,i} = \emptyset$ dokud P_i neodešle respektive nepřijme zprávu, kterou si sám odeslal.
- Nechť $-\sum_{i,j} = \{-m | m \in \sum_{i,j}\}$ a $+\sum_{j,i} = \{+m | m \in \sum_{j,i}\}$. δ_i je částečné zobrazení, $\delta_i : S_i \times (\langle \sum_{i,j} \rangle_{j \in I} \cup \langle + \sum_{j,i} \rangle_{j \in I}) \times I \rightarrow 2^{S_i}$. Dále $\delta_i(p, -m, j)$, je množina nových stavů do kterých může automat P_i přejít po odeslání zprávy typu m automatu P_j a $\delta_i(p, +m, j)$, je množina nových stavů do kterých může automat P_i přejít po přijetí zprávy typu m, která byla odeslána automatem P_j.
- s_{0i} je počáteční stav automatu P_i.

Moravec (2014) píše, že pro potřeby jeho práce byla uvedená definice čerpána z publikace Peng & Puroshothaman (1991) postačující a nezabývá se zápisem asynchronních komunikací pomocí zásobníků typu FIFO. Přijetí a odeslání zprávy ve stejném čase, jak píší (Peng & Puroshothaman, 1991), není potřeba využívat asynchronní komunikaci. Moravec (2014) dále uvádí publikaci (Brand & Zafiropulo, 1983) i přes to, že je starší třiceti let, je stále hlavním zdrojem v oblasti teorie komunikujících konečných automatů. Toto dokazuje, že autoři publikací Peng & Puroshothaman (1991) a Gouda et al. (1984), na publikaci autorů (Brand & Zafiropulo, 1983) odkazují.

Moravec (2014) píše, že pro grafické znázornění toků mezi externími entitami, inteními kroky zpracování a data-store elementy v rámci obchodního procesu slouží data flow diagramy (DFD) . Pro popis systému se zaměřením na toky dat v rámci systému, toky dat do a ze systému se používají DFD. V tomto pohledu jsou srovnatelné s vývojovými diagramy. DFD se od vývojových diagramů liší v zaměření,

a to tím, že DFD se zaobírají daty. Oproti tomu vývojové diagramy se zaobírají aktivitami a kontrolou řízení.

Mezi omezení DFD patří to, že jsou zaměřeny na data. DFD neposkytují žádné konstrukty pro podnikové procesy, které se skládají z popisu workflow, lidí, událostí a dalších prvků. V podobě DFD se jedná o statickou reprezentaci funkcionalit systému zahrnující datovou manipulaci. Protože DFD neobsahují konstrukty pro podnikové procesy, je obtížné je používat v analýze a řídících rozhodovacích prvcích.

Dalším typem diagramů jsou state transition diagramy (STD) , které jsou složeny z obdelníků a šipek, kdy obdélníky spojují a představují změny stavů, resp. přechody mezi stavy, které jsou reprezentovány obdélníky. ST diagramy vycházejí z analýzy a návrhu systému. Systém zpracovává úlohy v reálném čase a STD obsahují explicitní informace o časovém sledu událostí.Od DF diagramů se odlišují záznamem informací o čase. I přes záznam informací o čase se STD nezaměřují na workflow, řízení, rozhodování.

V specifikaci OMG (2001) se uvádí, že Unified Modeling Language (UML) roce 1997 přijala OMG (Object management Group) jako standardní jazyk pro specifikaci, konstrukci, vizualizaci a dokumentaci softwarových systémů. UML využívá širokou škálu diagramů, které lze rozdělit na:

- Diagramy po popis struktury, mezi ně patří například Class diagram.
- Diagramy po popis chování, mezi kterými je Use Case diagram, statechart, activity, state machine a interaction diagram.
- Class diagram slouží pro zachycení statického pohledu na systém, znázornění objektů a jejich vztahů v systému .
- Use Case diagram slouží pro zobrazení struktury systému z pohledu uživatele.

3.1.5 Přístupy pracující s UML

XIS je způsob generování z modelů (USE CASE, CLASS diagram, ...), které se převedou na informační systém jak pro desktop, tak pro mobilní zařízení.

RS lingo je způsob zápisu požadavků na informační systém, který je možné snadněji

validovat. Tento způsob zápisu ovšem neřeší převod slovního zápisu požadavku RS Lingo na zápis systematický (da Silva, 2003).

XIS přehled

XIS je projekt skládající se z jednotlivých elementů, které urychlují aktivity softwarového inženýrství. Jedná se o přístup k vývoji software, který je inspirovaný vybranými praktikami, jako jsou vysoko úrovňové modely nebo specifikace. Založen je na komponentových a generativních programovacích technikách (da Silva, 2003).

XIS následuje MDA filozofii. Platforma XIS je CASE nástroj podporující technické účastníky procesu vývoje softwaru podle XIS přístupu. XIS/UML profil je množina koherentního UML rozšíření, které umožňuje vysoko úrovňové vizuální modelování při návrhu Informačních systémů. XIS/XML jazyk je XML jazyk, který poskytuje textový, strukturovaný, srozumitelný a kompaktní způsob ke specifikování informačních systémů. Oproti XIS/UML slouží XIS/XML k textovému popisu informačních systémů (da Silva, 2003).

Proces tvorby informačního systému v XIS

Jako hlavní vstup slouží systémové požadavky (funkční, nefunkční a vývojové). Výstupem je množina artefaktů (zdrojové kódy, konfigurační nebo datové skripty). Hlavní postavou v procesu XIS je softwarový architekt, který má kritickou úlohu operací přístupu XIS. Dále je zodpovědný za následující úlohy: Booch et al. (1999) definuje srozumitelný a použitelný XIS/UML profil, Kruchten (2004), provádí podporu transformace, která definuje šablony pro převod z XMI na XIS/XML formát, a OMG (2001) definuje šablony softwarové architektury, které slouží k transformaci z XIS/XML specifikace na finální softwarové artefakty. Systémové požadavky (vytvořené například na setkání analytiků, klientů a koncových uživatelů) jsou základem dobrých výsledků. Analytici jsou zodpovědní za čitelné a odpovídající systémové modely. Správnost a kvalita těchto modelů je základem dobrých výsledků v dalších následných krocích.

Na úrovni XIS/UML není možné modelovat všechny systémové požadavky (například, obchodní pravidla nebo nefunkční požadavky), je tedy zapotřebí programátorů.

Programátoři doplní projekt pomocným kódem, jako jsou fasády, adaptéry, kontroléry a obchodní logika (Gamma et al., 1994), (*The Software Patterns Series*, 1996 - 2002). Dalšími účastníky projektu jsou testeři a integrátoři, kteří připravují a vykonávají testy k zachování systémové kvality (da Silva, 2014).

Hlavní principy přístupu XIS

XIS je založen na specifikaci modelů. Důležitými modely jsou XIS/UML profil a jazyková definice XIS/XML. XIS je dále založený na komponentách a nejvýznamnější postavou v projektu je softwarový architekt, jak již bylo popsáno výše. Mimo to je XIS také založen na technikách generování zdrojového kódu.

Projekt XIS je založen na specifikaci modelů.

UML profily a rozšiřující mechanismy

UML (Unified Modeling Language) je dobře známý OMG standard široce používaný v softwarovém průmyslu. Jedná se o modelovací jazyk k vizuální reprezentaci rozdílných perspektiv na informační systémy a k reprezentaci rozdílných úrovní abstrakce (Booch et al., 1999),(OMG, 2001),(OMG, 2003*b*).

UML umožňuje základní rozšíření bez speciálních požadavků (a také bez změny ve vnitřním meta-modelu) (OMG, 2001, 2003*b*, 1999). UML k tomuto poskytuje tři mechanismy: omezení, tag hodnoty (například metadata) a stereotypy (například metatypy). Tyto mechanismy dovolují:

- Představení nových elementů v modelu ke zvýšení srozumitelnosti.
- Definování položek, které rozšíří základní UML metamodel.
- Definování rozšíření, která jsou spojena s programovacím jazykem nebo specifickým vývojovým procesem.
- Přiřazení kontrolní sémantické informace k určitým elementům.

  UML profil je ve své podstatě jméno dané předefinované koherentní skupině stereotypů, tagů a omezení, které poskytují specializaci a konfiguraci UML pro specifickou aplikační doménu nebo specifický vývojový proces (OMG, 1999).

XIS/UML Profil

XIS/UML Profil je koherentní skupinou rozšíření UML, která dovoluje modelovat informační systémy podle přístupu XIS. Tento profil je nastaven podle dobře známých praktik inspirovanými publikacemi Krasner & Pope (1988), *The Software Patterns Series* (1996 - 2002),Juric et al. (2006). V literatuře Krasner & Pope (1988), *The Software Patterns Series* (1996 - 2002),Juric et al. (2006) je popisováno:

- Softwarová architektura je MVC (model-view-control).
- Business entity je doména.
- Předdefinované workflow uživatelského rozhraní (OMG, 2003*b*) a obecné workflow uživatelského rozhraní, například vysokoúrovňové jazyky jako jsou UIML (UIML.org, 2000; OMG, 2002) podnikový procesní tok a BPEL4WS (Ouvans et al., 2006).

XIS/UML profil umožňuje modelování informačního systému podle MVC architektury, založené na čtyřech doplňujících se modelech. Konkrétně jde o doménový model (Model), model podnikových entit (BusinessEntity), model procesní tok (Controller) a modely view (View). Tyto čtyři modely by měli být organizované v rozlišných UML balících s příslušnými závislostmi, kde balík View je závislý na balíku Controller a BusinessEntity, a balík BusinessEntity je závislý na balíku Model (da Silva, 2014).

Model entity (Model) Součástí doménového modelu jsou třídy a vztahy odpovídající entitám, které jsou rozpoznány během průzkumu problému domény (da Silva, 2003).

Model podnikových entit (BusinessEntity) Základem modelu podnikových entit je definice podnikových entit, které jsou fakticky sémantickou agregací několika entit definovaných v doménovém modelu. Tyto podnikové entity poskytují nejvyšší úroveň ve smyslu granularity podnikových entit, nebo interakci konečného uživatele v porovnání s ostatními modely.

Některé podnikové entity odpovídají jedna na jednu podle entit doménového modelu, zatímco některé odpovídají několika entitám. Pokud entity odpovídají několika,

je vhodné přidání sémantické závislosti k vyhodnocenému vztahu. Toto slouží k objasnění role nebo pořadí role entity ve vztahu k podnikové entitě. XIS podporuje sémantiku pomocí dvou hodnot tagů, konkrétně: role (hodnoty: master, detail, lookup) a index (používá se ke specifikaci pořadí). Pro komplexní podnikové entity, hodnota tagu role umožňuje specifikovat master-detail nebo lookup vztah. Tento vztah je velmi vhodný pro techniky generování kódu, částečně pro SQL skript (v návaznosti na garanci referenční integrity mezi různými tabulkami) nebo pro generování uživatelského rozhraní (da Silva, 2014).

Model pracovního toku (Controller) Model pracovního toku je užitečný pro definování třídy odpovídající předdefinovanému pracovnímu toku uživatelského rozhraní. Pracovní tok uživatelského rozhraní je v detailu znázorněn specifickým stavovým diagramem (da Silva, 2003).

Model View (View) V modelu view jsou definovány spojení mezi podnikovými entitami a předefinovaným pracovním tokem uživatelského rozhraní. Podle MVC architektury jsou podnikové entity (M) zobrazovány a řízeny v rozličných view (V) přes různé pracovní toky uživatelského rozhraní (C) (da Silva, 2014).

XMI jazyk XMI (XML metadata interchange) je OMG standard pro reprezentaci metadat (xmi, w3). XMI specifikuje schéma k reprezentaci metadat v závislosti na UML metamodelu. Hlavním cílem XMI je poskytnout součinnost metadat (konkrétně modelů UML) nezávisle na platformě, jazyku, uložišti a CASE nástroji (da Silva, 2014).

XIS/XML jazyk Výhodou XMI je, že umožňuje specifikaci modelu UML ve formátu XML nezávisle na nástroji a uložišti. Nevýhodami XMI jsou velikost, členění a komplexnost nutná k reprezentaci informace. Tyto nevýhody nejsou velkým problémem, protože XMI kód je zpracováván pomocí výpočetních nástrojů. I přes toto je XIS/XML přidaná hodnota, která poskytuje strukturovanější, srozumitelnější a kompaktnější formát k popisu informace. XIS/XML poskytuje pro lidi uživatele jednodušší definování nebo krokování obsažené informace v textovém formátu (da Silva, 2003).

XIS je založeno na generativním programování Dalším principem obsaženým v projektu XIS jsou techniky generativního programování (Cleaveland, 2001*b*; Group, 2004). Technika generativního programování je spojena s dalšími technikami, například založená na specifikaci modelů, komponentově založená a s orientací na architekturu. Technika generativního programování transformuje systémové specifikace na koncové softwarové artefakty a bere v potaz danou architekturu.

Přístup XIS teoreticky poskytuje produkci kódu v rozlišných programovacích jazycích v závislosti na softwarové architektuře, jak je popsáno v MDA filozofii. Automatický generativní XIS proces se skládá z následného použití tří transformací, které lze formalizovat do

$$IS = T3(T2(T1(XIS/UML - Model, IS/UML - Profile), XIS/XSL), SoftArch). \tag{3.23}$$

Nebo samostatně XIS/UML-Model (definováno přes nástroj CASE a založeno na XIS/UML profilu),

$$\begin{aligned} XMI - Model &= T1(XIS/UML - Model, XIS/UML - Profil) \\ XIS/XML - Model &= T2(XMI - Model, XIS/XSL) \\ IS(Finálnísystém) &= T3(XIS/XML - Model, SoftArch) \end{aligned} \tag{3.24}$$

kde jednotlivé zkratky znamenají, že XIS/UML-Model je shodný s originálním modelem specifikovaným s XIS/UML profilem. T1 je první transformace. T1 transformuje XIS/UML-Model na ekvivalentní model v XMI formátu. T2 je druhá transformace. T2 transformuje model specifikovaný v XMI formátu na ekvivalentní model v XIS/XML formátu. T2 je vykonávána pomocí XSLT parseru a je založena na vyvinutém XIS/XSL skriptu. T3 je třetí transformace. T3 vytváří automaticky několik artefaktů z modelů specifikovaných v XIS/XML a bere v potaz danou softwarovou architekturu (SoftArch). Generátor generického kódu se nazývá XIS-Generátor, který je klíčovým prvkem T3 transformace. IS je koncovým systémem, který se skládá z několika integrovaných artefaktů, jako jsou zdrojový kód, konfi-

gurace, skripty SQL, HTML, obrázky a dokumentace (da Silva, 2003).

Komplexní T3 transformace se skládá z generativního iterativního a inkrementálního přístupu, podpory sdíleného repositáře a integrace vygenerovaného a nevygenerovaného kódu (da Silva, 2003).

Iterativní a inkrementální přístup

Generativní přístup by měl být iterativní a inkrementální. Dobře známým problémem u generativního přístupu je omezení jeho rozsahu, protože běžně využívá strategii tzv. hrubé síly. To znamená, že pokud se objeví nový požadavek, je nutné všechny vytvořené artefakty znovu vygenerovat. Pro malé systémy je tento přístup možný. S rostoucím počtem systémů, které jsou komplexní a rozsáhlé, se stává tento přístup tzv. hrubé síly nepraktický. Berou-li se v potaz tyto požadavky (komplexnost a rozsáhlost systémů), je doporučován přístup založený na inkrementálním a iterativním modelu (da Silva, 2014).

Specifické generativní kroky jsou součástí generativního procesu. Pro každý generativní krok jsou specifikovány artefakty, které budou generovány (da Silva, 2014).

Podpora sdíleného repositáře

Transformace T3 se nevykonává přímo z XIS/XML specifikace. Proces T3 je založen na informaci uložené v XIS úložišti. T3 lze vidět jako pár dvou podprocesy. Jednoduchý proces (T3.1) na obsazení XIS repositáře informacemi definovanými podle XIS/XML specifikace. Komplexní proces (T3.2) slouží pro automatické generování různých softwarových artefaktů (jako například kód Java, JSP, HTML, ant-build, XML a skripty SQL) spojených do specifické aplikace. Tento proces je založený na informacích z XIS repositáře a na zvolené architektuře. XIS repositář je podporován relační databází (aktuálně MS-SQL Server) s komplexním datovým modelem s ohledem na řízení všech odpovídajících elementů (da Silva, 2003).

- Elementy XIS metamodelu (například elementy definované v XIS/UML profilu nebo XIS/XML úrovně), jako například entity, atributy, výčty, vztahy, pracovní toky, přechody, události nebo view (da Silva, 2003).

- Elementy podporující samotný generující proces (například T3.2), jako je architektura, kontext, aplikace, generativní kroky, generativní proces, generativní možnosti nebo softwarové artefakty (da Silva, 2003).

Integrace vygenerovaného a nevygenerovaného kódu

Problém může vyvstat při integraci vygenerovaného a nevygenerovaného kódu. Nemusí být jednoduché rozumět vygenerovanému kódu, a to např. z důvodu porozumění detailů vygenerovaného kódu. Ve většině případů je možné použít generátor, jako black box framework, a potom není nutné číst jeho zdrojový kód.

3.1.6 RSLingo

Inženýrství požadavků se skládá ze sedmi kroků, jak uvádí Sommerville (2010):

- Počátek požadavků
- Identifikování požadavků
- Analýza požadavků
- Specifikace požadavků
- Modelování systému
- Validace požadavků
- Řízení požadavků

Inženýrství požadavků se snaží o poskytování sdílené vize a porozumění mezi obchodními a technickými účastníky vyvíjeného systému. Nepříznivé důsledky nerespektování důležitosti počátečních aktivit, které jsou pokryty inženýrstvím požadavků, jsou dobře známy (El Emam & Koru, 2008), (Davis, 2013). Specifikace systémových požadavků, specifikace softwarových požadavků nebo pouze specifikace požadavků jsou dokument, který technicky popisuje softwarový systém (Sommerville & Sawyer, 1997), (Pohl, 2010), (Robertson & Robertson, 2012). Specifikace požadavků je používána v různých fázích životního cyklu projektu a napomáhá sdílení vize systému mezi hlavními účastníky, usnadňuje komunikaci a celkové řízení projektu

a také procesu vývoje systému. Každý by měl být schopný komunikovat běžným a přirozeným jazykem k dosažení efektivní komunikace. Přirozený jazyk je flexibilní, universální a lidé jsou sběhlí při komunikaci mezi sebou. Přirozený jazyk má minimální odpor k přijetí pro techniku dokumentace požadavků (Pohl, 2010), (Robertson & Robertson, 2012). Přesto, že přirozený jazyk je nejběžnější a je preferovaný při reprezentaci požadavků (Kovitz, 1999), také s sebou přináší mnoho kvalitativních nedostatků, jako je nepřesnost, nedůslednost, neúplnost a neurčitost (Pohl, 2010), (Robertson & Robertson, 2012). Neúplnost a nedůslednost jsou nejnáročnější na množství informací, pokud není použit vhodný nástroj, například na jazykovou analýzu požadavků. Na druhé straně dvojznačnost a nepřesnost nelze odstranit bez lidského ověření (Committee & Board, 1998).

Toto lze řešit pomocí přístupu, který představuje RSLingo (de Almeida Ferreira & Silva, 2012), (de Almeida Ferreira & Rodrigues da Silva, 2013). Přístup RSLingo slouží pro formální specifikaci softwarových požadavků, který používá odlehčené zpracování přirozeného jazyka (NLP) pro transformaci neformálních požadavků (Bird et al., 2009). Slouží pro překlad neformálních požadavků z přirozeného jazyka napsaného obchodními účastníky do formální reprezentace jazyka určeného pro inženýrství požadavků. Oproti tomu, ostatní přístupy k inženýrství požadavků využívají grafické modelovací nástroje, jako je například UML nebo BORM. RSLingo dovoluje obchodním účastníkům aktivně zasahovat do procesu požadavků pomocí přirozeného jazyka. Přístup RSLingo poskytuje techniku pro tento účel.

- RSL-PL jazyk pro definování jazykových vzorů, které se často opakují ve specifikaci požadavků napsaných v přirozeném jazyku.
- RSL-RSL jazyk pokrývá většinu požadavků formální specifikace.

Moravec (2014) uvádí rozmanitosti různých jazyků a prostředků sloužících pro popis softwaru. Tato práce bude z tohoto důvodu zaměřena pouze na notaci BPMN a metodu BORM. Tím, že existuje velká rozmanitost jazyků a prostředků pro popis softwaru a každý jazyk a prostředek má vlastní způsob analýzy a návrhu, můžou nastat problémy jazykového propojení. V případě UML, které pokrývá celý proces od analýzy až po softwarové komponenty a datbázové schéma, jde cestou univerzálnosti, a tím zaplňuje mezeru propojení jazyků.

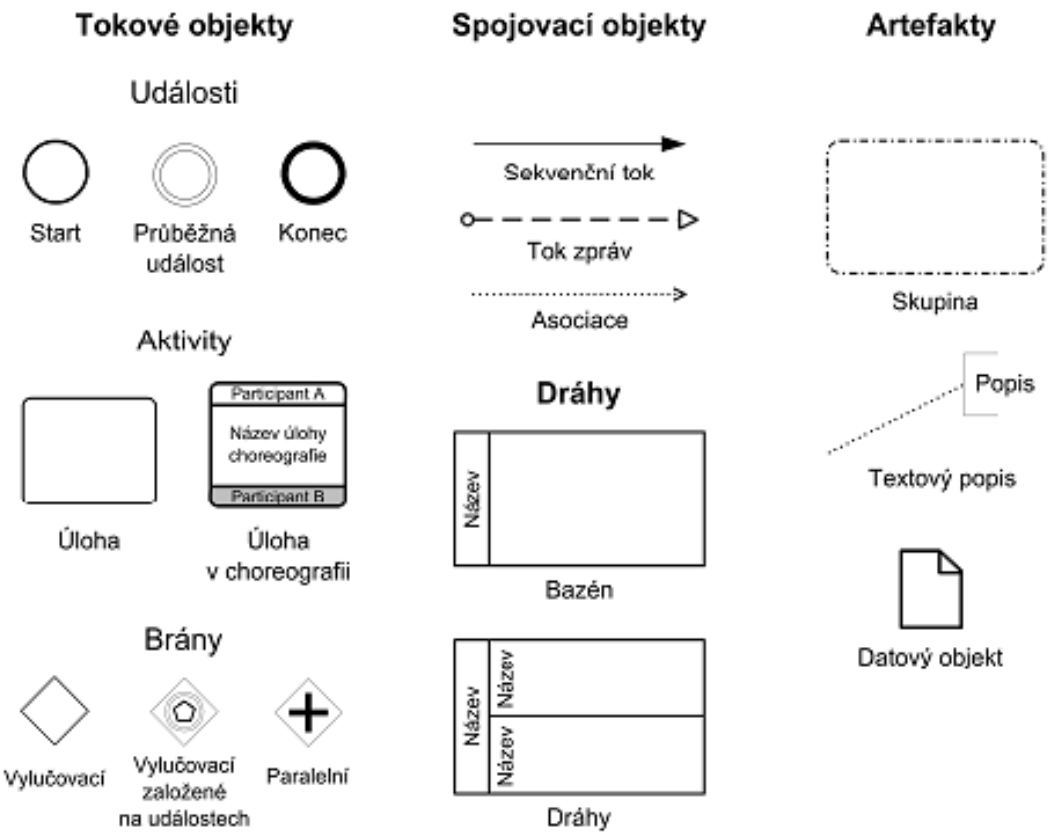
3.2 Lidsky čitelný zápis modelu

3.2.1 Business Process Model And Notation

Moravec (2014) píše, že během několika posledních let došlo ke značnému rozvoji vykonávacích jazyků pro řízení podnikových procesů. Jazyky jsou založeny na XML . BPMN také umožňuje převod na vykonávací jazyky. Převodem na vykonávací jazyk je umožněna vizualizace procesů. Tímto jsou rozšířeny možnosti notace a je možné modelovat obchodní procesy. Konkrétně jde modelovat komplexní procesy B2B. Procesy B2B jsou zaměřeny na komunikaci mezi podniky. BPMN bylo vytvořeno s cílem zachycení i velmi složitých podnikových procesů, a zároveň je to srozumitelný a jednoduchý mechanizmus na zachycení procesů srozumitelný a jednoduchý.

Požadavky na jednoduchost, srozumitelnost a zachycení velmi složitých procesů byly splněny díky tomu, že grafické elementy, ze kterých se skládá notace, byly rozděleny do pěti základních kategorií. Omezené množství základních kategorií umožňuje snadné pochopení a dobrou orientaci uživatele. Každá základní kategorie je složena z více dalších specifických elementů, jak je možné vidět na obrázku 3.3 na straně 36. Toto umožňuje vytváření komplexních a detailních procesních modelů. V každé kategorii jsou elementy znázorněny s podobným grafickým základem, tím je zachována pochopitelnost a čitelnost pro uživatele. Uživatele lze rozdělit na koncové, vlastníky a účastníky. Model BPMN a jeho formální analýza je popsána v článku (Lam, 2010). BPMN specifikace je popsána v publikacích od autorů Chinosi & Trombetta (2012), OMG (2006), Řepa (2007), OMG (2012), kde uváděno členění na pět kategorií grafických elementů procesu:

- Artefakty
- Data
- Dráhy
- Spojovací objekty
- Tokové objekty



Obrázek 3.3: Základní grafické elementy notace BPMN - zpracování Moravec (2014) podle OMG (2012)

Artefakty

Doplňující informace jsou zahrnuty do procesu pomocí artefaktů. Artefakty se použijí v případě, když nebylo možné jejich informaci zahrnout do procesu jiným způsobem. Artefakty rozšiřují informace a pochopení o samotném procesu.

V BPMN jsou definovány dva typy artefaktů Group (Skupina) a Text Annotation (Textový popis). Je možné vytvořit rozšiřující artefakty, ale tyto nové artefakty nesmějí být v rozporu s již vytvořenými artefakty a elementy, které jsou definovány v notaci.

Skupina je vytvořena grafickými elementy z totožné kategorie. Takto uživatelem vytvořené kategorie se využívají k analytickým účelům nebo dokumentaci. Sekvenční toky neovlivňují uspořádání skupiny (OMG, 2012).

Textový popis

Návrháři procesů používají textový popis pro doplňení diagramů o užitečné a podstatné informace pro koncové uživatele (OMG, 2012).

Data

Datové objekty se používají pro modelování dat. Informace, které obsahují datové objekty, jsou z vykonávání aktivit. Životní cyklus procesu nebo podprocesu je spojen s životním cyklem datového objektu, který je v něm obsažen (OMG, 2012).

Dráha

Bazén je rozdělen do drah. Dráha může být reprezentována jako role v podniku. Pokud bazén představuje jeden proces, potom dráhy jsou kategorie a organizují jednotlivé aktivity. Sekvenční toky mohou přecházet přes hranice drah, ale nemohou jít přes hranice bazénu (OMG, 2012).

Bazén

Bazén reprezentuje v BPMN účastníky procesu. Bazén odděluje činnosti od ostatních bazénů, jinak je možné posuzovat jej jako dráhu. Bazén je případem použití modelování B2B procesů.

Bazén v podobě černé skřínky je prázdný a jsou známy pouze vstupy a výstupy. Také může obsahovat procesy. Účastník procesu může být obecnější nebo konkrétnější. V obecném případě jde o role například výrobce, kupující, prodávající. V konkrétním případě jde o subjekt nebo osobu například společnost, účetní, ředitel.

Tokové zprávy zajišťují interakci mezi bazény a nelze je použít uvnitř bazénu. Sekvenční toky umožňují komunikovat uvnitř bazénu, ale nelze je použít mimo bazén pro komunikaci mezi bazény (OMG, 2012).

Spojovací objekty

Spojovací objekty je možné rozdělit na čtyři typy:

- Sekvenční toky
- Toky zpráv
- Asociace
- Datové asociace

Spojovací objekty se používají pro propojování tokových objektů navzájem nebo s ostatními prvky.

Sekvenční toky

Pořadí jednotlivých tokových objektů je určeno sekvenčními toky. Sekvenční toky určují pořadí v rámci procesu, jak budou tokové objekty vykonávány. Sekvenční tok nemůže překračovat mimo bazén a podproces. Sekvenční tok má svůj zdroj a cíl. Sekvenční tok může obsahovat podmínku. Podmíněný sekvenční tok je vykonán, pokud je podmínka splněna. V případě zdroje brány nebo aktivity se většinou vyskytují podmíněné sekvenční toky (OMG, 2012).

Toky zpráv

Účastníci procesu jsou reprezentováni jako samostatné bazény. Každý účastník má ekvivalent v bazénu. Při komunikaci mezi účastníky se využívá toku zpráv. Toky zpráv jsou definovány pro propojení mezi bazény a nejsou určeny pro komunikaci mezi tokovými objekty v rámci jednoho bazénu. Graficky se toky zpráv mohou spojovat bazény nebo tokové objekty uvnitř odlišných bazénů (OMG, 2012).

Asociace

Informace a artefakty jsou propojeny s ostatními elementy (často textové popisy) pomocí asociací (OMG, 2012).

Datové asociace

Pro přesun dat se využívají datové asociace. Data se přesouvají mezi vstupy, výstupy aktivit, procesů a datovými objekty. Data získáme z procesních datových vstupů nebo datových objektů a pomocí datových asociací jsou přeneseny do aktivit. Z aktivit jsou přeneseny opět pomocí datových asociací do procesních datových výstupů nebo datových objektů (OMG, 2012).

Tokové objekty

Moravec (2014) píše, že tokové objekty jsou základní grafické prvky. Tokové objekty definují chování obchodního procesu. Spojovací objekty propojují navzájem jednotlivé tokové objekty. Druhy tokových objektů jsou (OMG, 2012):

- Aktivity
- Brány (Gateways)
- Události

Události

Chandy (2006) definuje událost jako významnou změnu stavu. Intuitivně lze v průběhu procesu označit za událost něco, co se stane. Toky uvnitř modelu procesu jsou ovlivněny událostmi. Příčina nebo důsledek je obvykle označena událostmi. Kružnice je symbol pro událost. Do symbolu kružnice se doplňují další značky, které značí odlišné příčiny (spouštěče) a následné výsledky událostí. Rozlišujeme tři typy událostí (OMG, 2012):

- Start
- Průběžná událost
- Konec

Jednotlivé události jsou odlišeny podle toho, ve které fázi procesu je nacházejí a působí na tok procesu.

Událost typu start

definuje začátek procesu. Elementem typu start začíná tok procesu, pokud jde o sekvenci toků. Událost typu start je na začátku a nepředchází ji jiná událost. Událost typu start je možné spouštět pomocí specifických vlivů. Tyto specifické vlivy mají svoje značky, jak je zobrazeno v obrázku 3.3 na straně 36. V případě kdy je kružnice prázdná, jde o obecný začátek procesu, který není jasně definovaný (OMG, 2012).

Průběžné události

patří mezi začátek procesu a konec procesu. Průběžné události nemohou proces spustit ani ukončit, ale ovlivňují tok procesu. Sledování a kompenzace výjimek, příjem a odesílání zpráv, očekávání zpoždění, na tato místa je možné umístit průběžné události. Událost je možné připojit k aktivitě, potom vyjadřuje událost spuštění nebo přerušení aktivity (OMG, 2012).

Událost typu konec

definuje ukončení procesu. Analogicky k události typu start jsou specifické vlivy, které způsobí ukončení procesu (OMG, 2012).

Aktivity

Aktivity jsou prováděné práce v rámci procesu. Aktivita je konkrétní provedená práce. Rozdělují na složené (podprocesy) a jednoduché, nebo také nazývané atomické (úlohy) (OMG, 2012).

- Úloha
- Pod proces

Úloha

je jednoduchá (atomická) aktivita uvnitř procesu. Pokud úlohu není možné nebo nutné modelovat detailněji, je použita aktivita typu úloha (OMG, 2012).

Podproces

je skládaná aktivita a je součástí procesu. Podproces v diagramu je zobrazen v neotevřené podobě. Podproces je možné otevřít a zobrazit detailní pohled. Sekvenční toky uvnitř podprocesu jsou omezeny právě na podproces a nemohou zasahovat vně podproces (OMG, 2012).

Brány

Brány jsou řídící elementy sekvenčního toku. Používají se k rozdělení a slučování. V diagramu jsou vyobrazeny pomocí symbolu prázdného kosočtverce. Prázdný kosočtverec může obsahovat další specifické symboly, které dále určují chování brány (OMG, 2012).

- Vylučovací brány
- Vylučovací brány založené na datech
- Vylučovací brány založené na událostech
- Paralelní brány

Vylučovací brány

se dále dělí na vylučovací brány založené na datech a událostech. Větvení toku procesu pomocí vylučovacích bran je možné buď pomocí dat nebo událostí. Tok procesu pomocí vylučovacích bran je možné rozdělit na dvě i více cest. Výběr cesty je založen na konkrétních datech nebo události, ale vždy je vybrána pouze jedna cesta (OMG, 2012).

Vylučovací brána založená na datech

znamená, že dojde k rozdělení na cesty pomocí podmínky. Podmínka je řešena formou otázky, na kterou je možné odpovědět a množina odpovědí je úplně určena. Každá odpověď určuje právě jednu větev (cestu) procesu (OMG, 2012).

Vylučovací brány založené na událostech

jsou místa, kde dojde k rozdělení (větvení) procesu na základě události. Aktivováním události se vybírá větev, která bude použita pro proces. V diagramu jsou umístěny události hned za branou v jednotlivých cestách (OMG, 2012).

Paralelní brány

jsou odlišné od vylučovacích bran. Používají se pro paralelní toky, pro souběžné cesty v procesu, které je nutné vytvořit a synchronizovat. Synchronizace je založena na tom, že souběžné cesty se mohou provádět současně. Na slučovací paralelní bráně dojde k čekání než se dokončí všechny vstupní cesty (toky), které do této brány vedou. Na výstupu rozdělovací paralelní brány je rozdělen tok procesu na souběžné toky. V diagramu je zobrazena paralelní brána jako kosočtverec s vnitřním specifickým symbolem plus (+) (OMG, 2012).

3.2.2 Business Object Relation Modeling

Korthaus (1998) popisuje postupy použití objektově orientovaného přístupu, výhody a možnosti pro modelování obchodních procesů. V kontextu obchodních procesů podrobně rozebírá objektový přístup a jeho základní principy. Popis obchodních procesů, pomocí objektově orientovaného přístupu byl základem, spolu s propojením konečných automatů a objektově orientovaného přístupu, pro vznik metody BORM .

Moravec (2014) uvádí, že kniha autorů Shlaer & Mellor (1992) byla první zmínkou popisující možné propojení konečných automatů a objektově orientovaného přístupu.

Metoda BORM je popsána v publikacích od autorů Knott et al. (2006), Polák et al. (2003), Knott et al. (2003).

Od roku 1993 byla vyvíjena metoda BORM. Metodika pro objektově orientovaný software založený na objektově orientovaných programovacích jazycích a nerelačních objektových databázích bylo původním cílem metody BORM. Definované techniky v průběhu vývoje metody BORM se ukázali jako obecně použitelné pro modelování procesů. Tyto techniky je možné použít při tvorbě softwaru. Techniky z metody BORM je možné použít pro modelování obchodních procesů a k analýze požadavků na projektovaný systém.

Metoda BORM je používána ve firmě Deloitte&Touche Czech Republic and Central Europe. Touto firmou byl vývoj podporován od roku 1996.

Merunka (2008) píše, že charakteristické vlastnosti metody BORM jsou:

- Metoda BORM je navržena pro celé spektrum fází vývoje softwaru. Prioritou v metodě BORM je rozpoznání objektů v řešeně doméně problému v úvodních fázích projektu. Použití je možné i v doméně problému, která přímo nesouvisí se softwarem. Techniky z metody BORM se používají pro modelování procesů. Tyto samostatné techniky lze použít i bez následné implementace softwaru. Metoda BORM lze použít pro namodelování procesů a následnou optimalizaci problémové domény. Optimalizace úzce souvisí se zvýšením CMMI (Capability Maturity Model Integration) procesu.
- Životní cyklus metody BORM je rozdělen na více fází. V každé jednotlivé fázi je využita omezená sada pojmů. Transformace pojmů je předpokladem průběhu projektování. Pojmy agregace a dědičnost se nepoužívají v analytické fázi, ale až ve fázi implementační. Ve fázi analýzy se používají pojmy asociace, přechod nebo stav. Od analýzy po implementaci je rozšiřován detail v modelu a během životního cyklu je model transformován.
- V metodě BORM je každý pojem reprezentován shodnými symboly bez ohledu na to, jestli se jedná např. o diagramy datové struktury nebo komunikaci mezi objekty. Metoda BORM používá pro znázorňování konceptuálních a softwarových pojmů většinu symbolů shodně s jazykem UML, ale dovoluje v jednom

diagramu znázornit například posílání zpráv mezi metodami různých objektů v různých stavech. Tento přístup dovoluje vyjádřit konzistentním způsobem některé žádoucí detaily softwarové konstrukce, které lze výhodně aplikovat především při návrhu pro čistě objektově orientované programovací jazyky. Tento originální způsob nahrazuje tvorbu více od sebe oddělených třídních, stavových a kolaboračních diagramů, a také dovoluje zobrazit větší množství spolu souvisejících informací. Samostatné stavové či interakční diagramy jsou však samozřejmě v metodě BORM také používány.

Při řešení projektu podle zásad metody BORM lze rozlišit dvě hlavní fáze: fázi expanze a fázi konsolidace. Fáze expanze začíná analýzou modelu obchodních objektů. Dochází zde ke hromadění informací potřebných pro vytvoření aplikace.

Stadium expanze končí s dokončením analytického konceptuálního modelu, který na logické úrovni reprezentuje požadované zadání a v abstraktní podobě popisuje jeho řešení. Fáze konsolidace, tedy fáze od konceptuálního modelu až k finálnímu systému složenému ze softwarových objektů, se označují jako stadium konsolidace. Je tomu tak proto, že v těchto etapách se model, který je produktem předchozí expanze, postupně stává fungujícím programem. To znamená, že na nějakou myšlenkovou ”expanzi”zadání zde již není prostor ani čas.

V tomto stadiu se také počítá s tím, že od některých idejí z expanzního stadia bude třeba upustit vzhledem k časovým, kapacitním, implementačním nebo i intelektuálním schopnostem - odtud tedy název tohoto stadia. (Takto odstraněná informace však může být v budoucnu základem analýzy nové verze systému.)

Umění rozlišit a vyváženě řídit expanzi a konsolidaci je klíčovým faktorem úspěchu softwarových projektů. Samotný iterativní model totiž nezkušeného manažera může dovést k neúměrnému počtu iterací s dlouhými expanzemi. Konsolidace se potom odbývá a výsledný produkt nemá potřebnou kvalitu, jak popisuje Merunka (2008).

Metoda BORM má Knott et al. (2003) následující výhody, které jsou uvedeny v tabulce 3.1 na straně 45.

Tabulka 3.1: Výhody metody BORM dle autorů publikace (Knott et al., 2003)

Pořadí	Výhoda
1.	BORM je založen na předpokladu, že každému vývoji softwarového systému musí předcházet modelování obchodních procesů, jejichž vykonávání má vyvíjený systém podporovat. Díky tomu poskytuje BORM konzistentní přístup a notaci pro analýzu a návrh podnikového prostředí, které je zdrojem požadavků na systém a dále poskytuje nástroj pro analýzu a návrh softwaru.
2.	BORM navazuje na procesně orientovaný přístup v kombinaci s čistě objektově orientovaným paradigmatem, který se ukázal být výhodný pro vývoj softwaru. Procesně orientovaný přístup vede ke komplexnější a rychlejší a analýze při řešení problému.
3.	Koncoví uživatelé, nebo obecně účastněné strany problémové domény, jsou schopni velmi rychle pochopit přístup a diagramy používané v BORM.
4.	Podle zkušeností z Deloitte & Touche je analytická fáze projektu v BORM, v porovnání s ARIS , přibližně třikrát až čtyřikrát rychlejší.
5.	Metodika je snadno pochopitelná pro analytiky i vývojáře, protože je založená postupné transformaci modelu, přičemž v každé fázi se pracuje pouze s omezenou podmnožinou pojmů.
6.	BORM vychází z čistě objektového konceptu, a proto dovoluje použít více druhů hierarchie objektů, než ostatní metody vývoje softwaru. BORM pracuje se vztahy mezi objekty, které nemají přímou implementaci v běžných programovacích jazycích, ale jsou užitečné pro konceptuální modelování.

Rozšíření metody BORM

Na práci Shlaer & Mellor (1992), která byla zmiňována v souvislosti s původní koncepcí metody BORM, navazuje Merunka (2012*a*), který popisuje objektově orientovaný přístup k modelování procesů, založený na fundamentálních principech konečných automatů.

Základním kamenem je stále metoda BORM, ale namísto procesních diagramů, využívajících prvky definované v rámci této metody, byl použit jazyk UML, rozšířený o nový druh vztahu. Autoři obou uváděných prací Shlaer & Mellor (1992) a Korthaus (1998) se shodují, že při tvorbě procesního modelu je důležité nejprve identifikovat stavy, ve kterých se jednotliví participanti nacházejí.

Následně je možné identifikovat chování, respektive aktivity, které je nutné vykonat proto, aby participant přešel z aktuálního stavu do stavu nového. Procesní diagramy v BORM mohou obsahovat komunikace mezi aktivitami. Aby bylo možné použít prvky jazyka UML, bylo nutné definovat nový druh vztahu (komunikace), což UML umožňuje. Tento nový druh vztahu je analogický s datovými spojeními a je ho možné považovat za obecné předávání zpráv, které se používá v dynamických modelech UML.

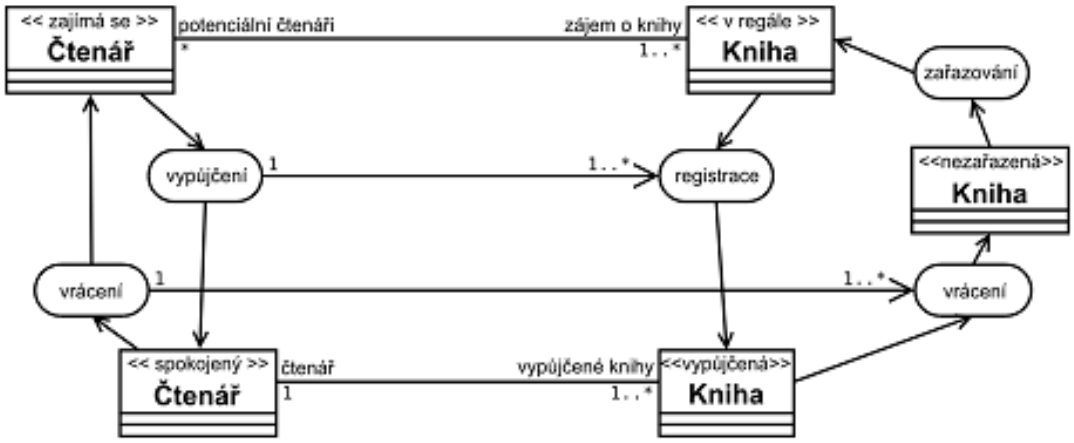
Nový přístup k diagramům v BORM využívá pro reprezentaci procesních modelů prvky UML, čímž sjednocuje objektové modely UML a techniky modelování obchodních procesů, jak je známe z BORM. Nespornou výhodou nového přístupu je relativně snadná transformace takto získaných modelů do jazyka UML nebo procesních modelů v BPMN, jak ukazuje ve své publikaci Merunka (2012*a*).

Možné výhody nového přístupu s využitím prvků UML viz obrázek 3.4 na straně 47, jak je uvádí autor publikace Merunka (2012*a*):

- Oproti diagramům v BORM, které jsou reprezentované pomocí výše zmiňované podoby jazyka UML, tyto diagramy BORM pokrývají BPMN a UML pouze část celého využitelného prostoru, který umožňuje objektově orientovaný přístup modelování.

  Participanty jsou v celé práci myšleni jak externí tak interní účastníci procesů.

- Nejdůležitějšími pojmy jsou stavy objektů. Aktivity představují ”pouze”způsob, jak přejít z jednoho stavu do stavu jiného. Při tvorbě procesních modelů



Obrázek 3.4: UML podoba procesních diagramů BORM - zpracování Moravec (2014) podle Merunka (2012*a*)

nebo návrhu softwarových komponent by se mělo začínat identifikací stavů, v kterých se jednotliví participanti v průběhu času mohou nacházet. Modelování procesů může být pak jednodušší, přesnější a méně závislé na chování participantů.

- Nový způsob reprezentace modelů také naznačuje zajímavé koncepční rozdíly mezi skládáním (kompozicí) a asociací. Kompozice představuje podrobnější pohled na stejný objekt, zatímco asociace představuje vztah mezi dvěma různými objekty. Toto rozlišení pojmů je velmi cenné zejména proto, že jsou tyto dva pojmy často směšovány a zaměňovány.

- Zavedení nového druhu vztahu - komunikace, který je zobecněním zasílání zpráv, jak jej známe z objektově orientovaného přístupu k modelování. Komunikace je propojení mezi aktivitami různých objektů. Stavy jednotlivých objektů mohou být také propojeny. Zavedený termín komunikace je symetrický k termínu asociace a může mít též definovanou kardinalitu. Asociace je informace o vztahu mezi dvěma objekty nebo jejich stavy, zatímco komunikace je analogický vztah mezi chováním (aktivitami) těchto objektů. Významný rozdíl mezi pojmem asociace a skládání může pomoci vyřešit problém konceptuálního a technického nesouladu v průběhu implementace softwaru.

3.2.3 Analýza procesního modelu pomocí OBA

Metoda OBA je iterativní technika sloužící k získávání strukturovaných podkladů ze zadání, nejčastěji na základě řízeného interview, pro potřeby konstrukce pr-

votního objektového modelu. Právě proto je velmi vhodná pro nasazení v počáteční fázi tvorby informačních systémů podle zásad metody BORM. Výstupy z OBA analýzy následně slouží ke konstrukci diagramů obchodních objektů. Metoda vznikla počátkem 90. let na základě zkušeností s aplikacemi různých technik JAD (Joint Application Design) a CRC (Class-Responsibility-Collaborator) pro potřeby objektové analýzy a návrhu a implementace v objektově orientovaných programovacích jazycích jak uvádí Merunka (2012*a*).

Objektově orientovaná analýza se snaží modelovat situaci vyjádřenou jako množinu vzájemně interagujících entit, kde má každá z entit jasně definovanou sadu chování a atributů (Shlaer & Mellor, 1992). Jak uvádí autoři Goldberg & Rubin (1995), existuje mnoho přístupů s podobným konceptem, ale odlišnou terminologií. Jinými slovy výsledky různých přístupů jsou velmi podobné, ale liší se v postupu, kterým je konečného výsledku dosaženo. Obecně se začíná u slovního popisu modelovaného problému. Dále se identifikují fyzické objekty, tedy v popisu problému se hledají podstatná jména, slovesa a přídavná jména. Na základě nalezených podstatných jmen se definují objekty, od sloves se odvozují rozhraní pro zprávy a přídavná jména slouží pro odvození logiky chování objektů. Následkem takového přístupu je silná preference fyzických objektů na úkor konceptuálních objektů, což může mít významný vliv na výsledek analýzy.

Pro malé systémy může být tento základní přístup postačující, což ovšem neplatí pro velká řešení, jak uvádí autoři Goldberg & Rubin (1995). Použití tohoto základního principu je podmíněno úplnou a formálně správnou specifikací problémové domény, která zvláště pro velké systémy není vždy k dispozici. Řešení nabízejí autoři Goldberg & Rubin (1995) v podobě OBA (Object Behavioral Analysis), která představuje efektivnější způsob, jak identifikovat objekty ze slovního popisu problému. Na první místo kladou autoři pochopení toho, co se v systému děje, tedy na chování systému. Tato chování je dále nutné přiřadit konkrétním částem systému a určit, kdo je iniciuje a kdo se na nich podílí. Určení iniciátorů a účastníků procesu napomáhá k pochopení různých rolí systému a určuje, které jeho části jsou odpovědné za poskytování služeb a řízení. Iniciátoři a účastníci, kteří mají významné role v rámci systému, jsou považováni za objekty a je jim přiřazena odpovědnost za chování odpovídající konkrétní části systému.

Metoda OBA se podle autorů (Goldberg & Rubin, 1995) snaží určit:

- Jaké role a odpovědnosti jsou nezbytné k vykonání určitého úkolu.
- Proč určitý objekt existuje.
- Proč jsou jednotlivé objekty propojeny tak, jak jsou.
- Jaké služby daný objekt poskytuje.
- Jak se objekt podílí na plnění funkčních požadavků na systém.

Cílem OBA je v první řadě pochopit verbální popis problému a problém zapsat jako interakce objektů. Tyto objekty plní systémové role tím, že si navzájem poskytují služby tak, aby naplnily požadavky na chování systému. Výsledek analýzy by měl být srozumitelný pro koncového uživatele a měl by sloužit jako podklad pro návrh konkrétní implementace. Musí zde být jasná návaznost na cíle a záměry definované v prvním kroku analýzy (Goldberg & Rubin, 1995).

Autoři OBA (Goldberg & Rubin, 1995) uvádějí rozdělení metody na následujících pět kroků:

- 1. Stanovení kontextu pro analýzu. První krok spočívá v identifikaci cílů a záměrů, nalezení vhodných zdrojů pro analýzu, identifikaci klíčových oblastí a aktivit a sestavení předběžného plánu průběhu analýzy. Jelikož se jedná o přípravnou fázi, která jde nad rámec analýzy samotné, lze ji vynechat. Nicméně poskytuje odrazový můstek pro další kroky. V průběhu prvního kroku může dojít k odhalení slabých míst v koncepci případně nekonzistence mezi záměry a cíli, což může mít dopad na výsledek analýzy.
- 2. Pochopení modelovaného problému na základě požadovaného chování. Druhým krokem je určit, jaké klíčové funkce by měl systém vykonávat, pro koho by je měl vykonávat a kdo by měl funkce vykonávat. Základem je vytvoření scénářů, které pokrývají všechny možné funkce systému. Někdy se tyto scénáře označují jako případy užití (Use Cases). Obvykle se podklady pro sestavení scénářů získávají z interview s uživateli a odborníky na danou problémovou doménu. Scénáře popisují modelovaný proces jako posloupnost elementárních činností, které na sebe navazují. Definuje podmínky nutné pro realizaci daného scénáře (precondition), účastníky (participanty) procesu, kteří se podílejí na realizaci scénáře. Jako poslední prvek obsahují scénáře výsledky procesu (postcondition). Všechny uvedené prvky jsou zachyceny v textové podobě a uspořádané

do tabulky. Výsledkem druhého kroku je pak sada scénářů, kde pro každý proces rozpoznaný v rámci interview je definován vlastní scénář.

- 3. Definice objektů.Cílem třetího kroku je definování objektů vykazujících chování, které přispívá k dokončení určitého úkolu. Každý participant nebo externí účastník procesu je potenciálním objektem majícím specifické vlastnosti. Takový objekt může obsahovat data, poskytovat služby respektive vykonávat aktivity, které slouží k transformacím dat uvnitř objektu a také k jeho komunikaci s ostatními objekty. Pro zachycení vlastností objektu slouží upravená varianta CRC karet, které se v OBA nazývají Modelové karty. Výstupem třetího kroku je sada modelových karet, které pro daný objekt definují jeho atributy, služby a spolupracující objekty.

- 4. Klasifikace objektů a identifikace vzájemných vazeb. Čtvrtý krok rozšiřuje obsah modelových karet z předchozího kroku o bližší specifikaci vzájemných vztahů jednotlivých objektů. Vzájemné vztahy jsou rozšířeny o případy, kdy objekt odešle jinému objektu zprávu za účelem získání nebo předání informací, případně si objekty předávají řízení procesu. Autoři metody OBA zavádějí následující tři reorganizační techniky, které slouží k identifikaci služeb a logických vlastností společných pro dva a více objektů:

  a) Abstrakce znamená mají-li dva nebo více objektů definované stejné služby nebo logické vlastnosti, je vytvořen nový objekt, kterému tyto vlastnosti nebo služby jsou přiřazeny. Zároveň jsou tyto vlastnosti nebo služby od původních objektů odebrány.

  b) Specializace se v mnohém podobá abstrakci. Postup je velmi podobný. V případech kdy dva nebo více objektů poskytují stejné služby nebo mají stejné logické vlastnosti, tyto jsou od původních objektů odejmuty a je vytvořen objekt nový, kterému jsou tyto služby nebo logické vlastnosti přiřazeny. Na rozdíl od abstrakce, v případě specializace je přiřazen původním objektům atribut obsahující nově vytvořený objekt, kterému byly odejmuté vlastnosti nebo služby přiřazeny.

  c) Faktorizace je proces, kdy každý účastník procesu - tedy objekt, který se účastní více scénářů a který poskytuje velký počet služeb, lze rozdělit na více objektů. Každému z nově vytvořených objektů je pak přiřazena podmnožina atributů nebo služeb původního objektu. Původní objekt pak lze vyřadit z množiny objektů. Cílem je vytvořit z jednoho univerzálního objektu více specializovaných objektů.

- 5. Modelování dynamiky systému popisuje oproti předchozím krokům, které popisovaly strukturu systému staticky, tedy v jednom místě a čase, aspekty systému, které se v čase mění. Všechny dosud identifikované a nově vytvořené objekty mohou měnit v čase svůj stav na základě událostí, které vykonávají. Pro každý objekt je definována konečná množina stavů, ve kterých se v průběhu životního cyklu objekt může nacházet. Životní cyklus objektu vyjadřuje změny jeho stavu v průběhu času. Stavy jednotlivým objektům přidělujeme na základě scénářů, definovaných v předchozích krocích. Pokud zjistíme, že je nutné přiřadit danému objektu další stav, který nevyplývá ze scénářů, musíme se vrátit zpět ke kroku jedna a upravit konkrétní scénář. Dbáme na to, aby každý objekt byl v jednom okamžiku pouze v jednom stavu.

  OBA je jedním z nástrojů metody BORM. Jako součást BORM byla OBA oproti původním pěti krokům, jak je uvádějí autoři Goldberg & Rubin (1995), modifikována. Jednotlivé kroky použití OBA pro analýzu procesního modelu podle metody BORM, jak je uvádí autor publikace Merunka (2008), jsou následující:

- Rozpoznání procesů se na základě provedeného interview sestaví seznam požadovaných funkcí systému a klíčové objekty v systému. Jedná se o slovní respektive textové popisy procesů. Cílem tohoto kroku je nejen zahájit stavbu modelu, ale i vymezit zadání v rámci možného širšího kontextu řešeného problému.

- Rozpoznání plánování scénářů je detailní popis již rozpoznaných funkcí a popisy vlastností objektů. Na základě funkcí rozpoznaných v předchozím kroku se vytvoří seznam scénářů. Dále se u každého scénáře rozlišuje původ procesu, vlastní popis procesu, participující objekty a popis výsledku procesu. Scénáře jsou již strukturované a rozšiřují popis procesu (Merunka, 2008).

- Definování vztahů mezi objekty navzájem a mezi objekty a procesy pomocí modelových karet. V tomto kroku se pro každý rozpoznaný objekt z předchozího kroku vytvoří jeho modelová karta, která obsahuje jméno objektu, seznam aktivit objektu a s ním související seznam s modelovaným objektem spolupracujících objektů (Merunka, 2008).

- Při modelování procesů se pro každý rozpoznaný objekt s pomocí informací v tabulce scénářů a modelových kartách sestaví životní cyklus objektu jako sled jeho stavů a přechodů mezi těmito stavy v podobě procesního diagramu (Merunka, 2008).

- Při verifikaci a validaci se kontroluje shoda mezi diagramy, tabulkami a skutečnými požadavky na systém. K tomu slouží dva nástroje. Jedním z těchto nástrojů je datový model obsahující skutečná data, pomocí kterých lze prověřit správnost návrhu. Druhým nástrojem je simulátor procesů, který dovoluje procházet jednotlivé kroky procesu znázorněného diagramy a umožňuje tak odhalit zacyklení nebo případná uváznutí procesu (Merunka, 2008).

3.2.4 Procesní diagramy v BORM

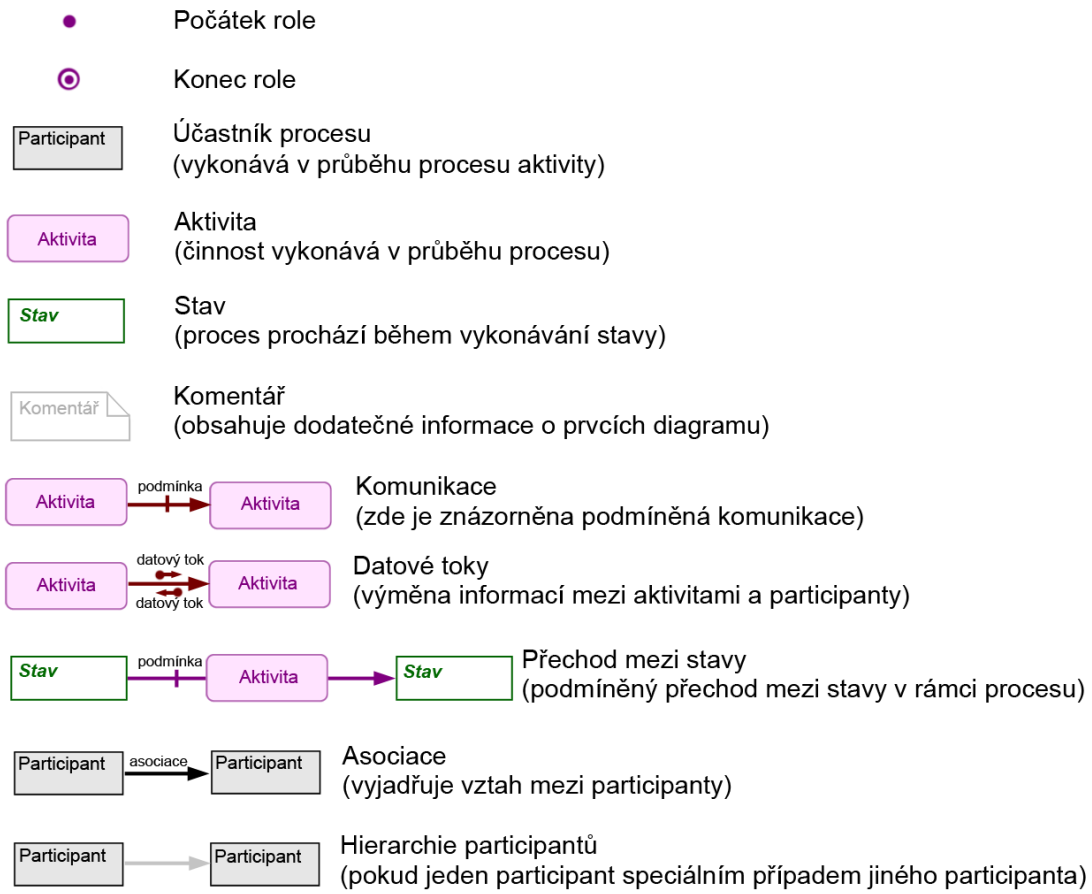
Diagram ORD (Object-Relationship-Diagram) byl vyvinut k vizuální reprezentaci informací o procesech a objektech získaných metodou OBA. Jedná se o jednoduchý diagram, který obsahuje jen malý počet pojmů a symbolů, které jsou plně postačující pro prvotní popis modelovaných procesů, a tím je použitelný i pro konzultace se zadavateli/zákazníky (Knott et al., 2000).

Jak uvádí dále Knott et al. (2000), ORD dovoluje modelovat jednotlivé procesy současně dvojím způsobem:

- Sekvence stavů a přechodů každého objektu, na které lze hledět jako na jednotlivé stavové diagramy, vyjadřují roli daného objektu v modelovaném procesu. Tento pohled slouží ke kontrole celkového modelovaného procesu například při interview.
- Sled komunikací mezi aktivitami různých objektů v různých stavech vyjadřuje průběh vlastního procesu. Celkový proces je tedy znázorněn jako propojení rolí objektů, které se tohoto procesu účastní. Hledíme-li na participující objekty se svými stavy a přechody jako na automaty, jedná se o zobrazení průběhu procesu metodou komunikace automatů mezi sebou, kde výstup jednoho objektu je vstupem pro jiný objekt.

Vzhledem k tomu, že modelovaný proces je konstruován jako propojení rolí (stavů a přechodů) účastnících se objektů, tak ORD dovoluje jednoduchým a nenásilným způsobem zachytit přesný průběh modelovaného procesu, čímž poskytuje také prostředky pro ověřování jeho správnosti. ORD není možné rozšířit o aktivitu, která by nenavazovala na nějaký již přítomný stav nebo nebyla vázána nějakou komunikací s jinou aktivitou. ORD je diagram, kde každý objekt (participant) je

modelován jako Mealyho automat. Při simulaci průběhu procesu v nástroji Craft-CASE se využívá synchronizace pomocí Petriho sítě. Tyto vlastnosti, které přímo vyplývají z použité teorie, jsou velmi dobře využitelné v interview, ve kterých se diagram sestavuje nebo verifikuje (Merunka, 2008).



Obrázek 3.5: Základní grafické elementy notace BORM - zpracování (Moravec, 2014)

Jak ukazuje obrázek 3.5 na stránce 53, základními prvky procesního diagramu jsou podle Moravec (2014), který vychází z Merunka (2012*a*):

- Participant reprezentuje účastníka procesu a je zobrazen jako orámovaný obdélník.
- Stav je obsažen v participantu, který může v čase měnit svůj stav na základě vykonaných činností. Stavy jsou reprezentovány obdélníkem.
- Činnost nebo aktivita představuje činnosti vykonávané participantem. Slouží pro přechod z jednoho stavu do jiného. Pro aktivity je použit obdélník se zaoblenými rohy.
- Komunikace propojují stavy a činnosti v rámci jednoho participanta nebo vyjadřují vzájemnou komunikací více participantů. Komunikace jsou znázorněny

šipkou. Součástí komunikace mohou být data, která jsou znázorněna menší šipkou spolu s identifikací dat.

3.2.5 Formalizace BORM ORD

Moravec (2014) píše při volbě vhodného formálního aparátu pro popis procesních diagramů v BORM budeme vycházet z práce (Merunka, 2008), který uvádí že procesní diagram v BORM je tvořen participanty, kde každý z nich je modelován jako Mealyho konečný automat. Jako vhodný formální prostředek pro popis procesního diagramu se tedy jeví použít Mealyho automat nebo obecně koncept z oblasti teorie konečných automatů.

Pro formální popis procesního diagramu v BORM by bylo možné použít například Petriho sítě. Během zpracovávání této práce se její autor zabýval možností transformace procesních diagramů v BORM na Petriho síť a její využití pro popis procesu. Možnosti využití Petriho sítí ve spojení s BORM byly prezentovány na konferenci Objekty 2010 (Moravec & Papík, 2010). Příspěvek mimo jiné poukazuje na možnost využití Petriho sítí, přesněji řečeno P/T Petriho sítí, jako nástroje pro validaci a simulaci procesů. Použití Petriho sítí pro modelování procesů je vhodné v případech, kdy máme relativně jednoduchý proces, protože u komplexních procesů může být model díky své velikosti nepřehledný. Nevýhodou užití Petriho sítí u rozsáhlých procesů je, že neposkytují komplexní pohled na modelovaný proces, protože je nutné tento proces rozdělit na více vzájemně propojených podprocesů. Vhodnou oblastí použití je konstrukce validátoru a simulátoru procesu. Petriho sítě dovolují stanovit i složité podmínky pro přechody do jednotlivých stavů procesů pomocí soustav rovnic což dovoluje mnohem přesněji řídit průběh procesu (Murata, 1989).

Jelikož se předpokládá, že každý participant je modelován jako Mealyho konečný automat a pro popis komunikací, které obsahují data nelze použít P/T Petriho sítě, dospěl autor této práce k závěru, že bude vhodnější použít pro formální popis ORD konečné automaty namísto P/T Petriho sítí. Nutno dodat, že existují Barevné Petriho sítě (Coloured Petri Nets), které umožňují definovat datové typy a modelovat komplexní manipulaci s daty (Jensen, 1987).

Popis ORD pomocí konečného automatu

Procesní diagramy v BORM, neboli ORD (Object Relation Diagram), jsou vizuální reprezentací informací o procesech a objektech získaných metodou OBA. Procesní diagram se skládá z participantů a případně jejich vzájemných komunikací. Každý participant je tvořen množinou stavů, aktivit, přechodů (komunikací).

Komunikace v rámci jednoho participanta je považována za přechod, neboť neobsahuje žádná data. Naopak komunikace mezi participanty může obsahovat data, a proto jsou to datové toky. Celý procesní diagram je reprezentován jako množina automatů, kde každý automat reprezentuje právě jednoho participanta (Moravec, 2014).

Popis chování participanta

Základní koncept konečného automatu, jak je popsán v kapitole 3.1.4 Mealyho automat na straně 21, je

Definice 3.25. každý participant **Pi** reprezentující jedinečnou modelovanou entitu definován jako uspořádaná pětice

$$P_i\langle S_i, I_i, \delta_i, s_i^0, s_i^e\rangle, \text{ kde:} \tag{3.26}$$

- S_i je konečná neprázdná množina všech stavů, v nichž se participant může nacházet.
- I_i je konečná neprázdná množina všech vstupů.
- δ_i reprezentuje vykonávané aktivity, tedy přechody mezi stavy; $\delta_i : I_i \times S_i \to S_i$.
- s_i^0 je počáteční stav procesu a platí, že $s_i^0 \in S_i$.
- s_i^e je koncový stav procesu a platí, že $s_i^e \in S_i$.

Participant začíná svoji činnost ze stavu s_i^0 a na základě uživatelského vstupu I_i a aktuálního stavu přechází do jiného, následujícího stavu. Pokud participant skončí ve stavu s_i^e můžeme říci, že P_i zpracoval slovo ze vstupu I_i^*. Je povoleno čtení

prázdného symbolu ϵ, takže participant může přejít do dalšího stavu, a to i bez zásahu uživatele.

V kapitole 3.1.4 Mealyho automat na straně 21 byl formálním způsobem popsán konečný automat jako uspořádaná pětice $A\langle S, A, \mu, s_0, s_f\rangle$. Na základě výše uvedeného popisu je ekvivalence mezi popisem chování participanta $P_i\langle S_i, I_i, \delta_i, s_i^0, s_i^e\rangle$ a popisem konečného automatu, jak je možné vidět v tabulce 3.2.

Participant		Konečný automat
S_i	$\Leftrightarrow$	S
I_i	$\Leftrightarrow$	A
$\delta_i : I_i \times S_i \to S_i$	$\Leftrightarrow$	$\mu : S \times A \to S$
$s_i^0; s_i^0 \in S_i$	$\Leftrightarrow$	$s_0; s_0 \in S$
$s_i^e; s_i^e \in S_i$	$\Leftrightarrow$	$s_f; s_f \in S$

Tabulka 3.2: Tabulka ekvivalence popisu participanta a konečného automatu (Moravec, 2014)

Proces s více nekomunikujícími participanty

Věta 3.26.1. Moravec (2014) píše, že byl popsán model pouze s jedním participantem. Model bude rozšířen o **N** vzájemně nekomunikujících participantů $P_1, ..., P_N$, jejichž činnost se bude simulovat zároveň. Předpokládá se, že vstupní symboly jsou pro každého participanta odlišné.

$$\cap_{i=1}^{n} I_i \subseteq \epsilon \tag{3.27}$$

Definice 3.28. Je definováno, že jeden konečný automat P_Σ, který bude kompozicí dílčích automatů reprezentujících jednotlivé participanty. Takový automat bude simulovat činnost všech participantů současně.

$$S = S_1 \times ... \times S_N, s_0 = \langle s_1^0, s_2^0, ..., s_N^0\rangle, s_e = \langle s_1^e, s_2^e, ..., s_N^e\rangle \tag{3.29}$$

$$I = \cup_{i=1}^{N} I_i. \tag{3.30}$$

Věta 3.30.1. Potom vnitřní stavy P_Σ jsou n-tice o délce **N**, kde **N** je počet všech participantů, složené z vnitřních stavů jednotlivých participantů. Obdobný přístup

je k počátečním a koncovým stavům. Předpokládá se, že vstupní symboly jsou pro každého participanta odlišné, což nám umožní relativně jednoduše definovat přechodovou funkci pro tento komponovaný automat s pomocí přechodových funkcí δ dílčích automatů.

$$\delta : I \times S \to S. \tag{3.31}$$

$$\delta(i, s_1, ..., s_N) = (s_1, ..., s_{j-1}, s^{'}, s_{j+1}, ..., s_N) | \exists j \in N, \delta_j(i, s_j) = s^{'}_j. \tag{3.32}$$

Výše uvedený popis je konečný pro participanty bez vzájemné komunikace. Tedy až doposud byl model tvořen jedním komponovaným automatem složeným z dílčích automatů, který každý odpovídá jednomu účastníkovi. Obchodní procesy nejsou tvořeny pouze izolovanými participanty. Obsahují také komunikace mezi participanty a stejně tak mohou obsahovat i komunikace napříč procesy. Komunikace je realizována zasíláním nebo přijímáním zpráv. Zprávy jsou zasílány v průběhu vykonávání činností, tedy v rámci aktivit procesu. Model komunikace použitý v procesních diagramech BORM předpokládá, že operace komunikace je dále nedělitelná, tedy atomická a probíhá mezi dvěma a více účastníky zároveň (Merunka, 2012*a*). Pokud není příjemce zprávy v příslušném stavu, který předpokládá přijetí zprávy, nelze komunikaci provést. Zaslání zprávy, tedy realizaci komunikace, lze považovat za činnost, která může být provedena pouze tehdy, když jsou všichni participanti v příslušných stavech. Realizace komunikace pak vede k současnému přechodu všech komunikujících participantů do nových stavů. Tato teorie nevystačí pouze se základní koncepcí konečného automatu. Koncept konečného automatu, který takové komunikace umožňuje, byl popsán v kapitole 3.1.4 Komunikující konečné automaty na straně 23. Přítomnost komunikací vyžaduje významnou změnu oproti výše uvedené definici participantů, ale dovolí popsat procesní diagram jako celek. Následující popis procesu ideově vychází ze základního pojetí konečného automatu, jak bylo aplikováno výše, ale obohacuje jej o prvky z modelu komunikujících konečných automatů, které jsou nezbytné pro zachycení vzájemných komunikací participantů.

Proces s více vzájemně komunikujícími participanty

Peng & Puroshothaman (1991) a Brand & Zafiropulo (1983) píší jako Moravec (2014), že syntézou formálního popisu Mealyho konečného automatu a vybraných prvků, které se týkají zasílání zpráv z formalizmu komunikujících konečných automatů, popsáno v kapitole 3.1.4 Komunikující konečné automaty na straně 23, bylo možné odvodit popis procesního diagramu, ve kterém spolu mohou jednotliví participanti navzájem komunikovat. Součástí komunikace mohou být také data, což by při použití pouze Mealyho konečného automatu nebylo možné. Procesní diagram reprezentující konkrétní proces lze definovat jako množinu participantů.

$$BP = P^i. \tag{3.33}$$

Definice 3.34. Každého participanta pak můžeme popsat jako uspořádanou šestici $P^i\langle S^i, -M^i, +M^i, f^i, g^i, s_1^i\rangle$ kde:

- S^i je konečná množina všech stavů.
- $-M^i$ je konečná množina všech odchozích zpráv.
- $+M^i$ je konečná množina všech příchozích zpráv.
- f^i reprezentuje vykonávané aktivity, tedy přechody mezi stavy. Přechodovou funkci pak lze definovat jako $f^i : S^i \times +M^i \to S^i$.
- g^i je výstupní funkce, kterou můžeme definovat jako $g^i : S^i \times +M^i \to -M^i$.
- s_1^i je počáteční stav participanta P^i , přičemž platí že $s_1^i \in S^i$.

Obdobně jako v případě nekomunikujícího participanta, komunikující participant začíná svoji činnost ze stavu s_1^i , který je jedním z konečné množiny všech stavů S^i. Přechody mezi jeho stavy jsou dány přechodovou funkcí f^i a závisejí na příchozích zprávách daných množinou $+M^i$. Na rozdíl od předchozího případu bylo nutné zavést, v souladu s definicí Mealyho konečného automatu, také množinu výstupů $-M^i$ participanta, která je dána výstupní funkcí g^i.

V kapitole 3.1.4 Mealyho automat na straně 21 byl formálním způsobem popsán Mealyho konečný automat jako uspořádaná šestice $M\langle S, s_0, A, B, \psi, \mu\rangle$. Na základě

výše uvedeného aparátu můžeme nalézt ekvivalenci mezi popisem chování komunikujícího participanta
$P^i\langle S^i, -M^i, +M^i, f^i, g^i, s_1^i\rangle$ a popisem Mealyho konečného automatu.

Lemma 3.34.1. Tato ekvivalence umožňuje považovat každého participanta procesu za Mealyho konečný automat.

Komunikující participant		Mealyho konečný automat
S^i	$\Leftrightarrow$	S
$-M^i$	$\Leftrightarrow$	A
$+M^i$	$\Leftrightarrow$	B
$f^i : S^i \times +M^i \to S^i$	$\Leftrightarrow$	$\psi(s0) : A \to (B \times S)$
$s_1^i; s_1^i \in S^i$	$\Leftrightarrow$	$s_0; s_0 \in S$
$g^i : S^i \times +M^i \to -M^i$	$\Leftrightarrow$	$\mu : S \times A \to B$

Tabulka 3.3: Ekvivalence popisu Mealyho konečného automatu a popisu komunikujícího participanta (Moravec, 2014)

Moravec (2014) analogicky jako Peng & Puroshothaman (1991) a Brand & Zafiropulo (1983) píše, že pro popis celého procesního diagramu ORD (Object Relation Diagram) BORM pomocí jednoho komponovaného konečného automatu je nutná podmínka, aby jednotliví participanti mohli navzájem komunikovat a každá komunikace může obsahovat data. Propojením mechanismu komunikujících konečných automatů, popsaných v kapitole 3.1.4 Komunikující konečné automaty na straně 23, a komunikujícího participanta, popsaného v této kapitole, lze popsat mechanismus zasílání zpráv v případě modifikované podoby Mealyho konečného automatu.

Definice 3.35. Předpokládejme, bez ztráty obecnosti, že participant nebude zasílat zprávu sám sobě.

$$+M^i \cap -M^i = \emptyset. \tag{3.36}$$

Definice 3.37. Množina všech zpráv (komunikací) i-tého participanta pak bude sjednocením množin všech odchozích a příchozích zpráv.

$$M^i = -M^i \cup +M^i. \tag{3.38}$$

Definice 3.39. Bez ztráty obecnosti se předpokládá, že v celém systému komunikací je právě jedno identické m_i, tedy že každá zpráva má právě jednoho příjemce a jednoho odesílatele.

$$\forall m_1 \in M^1, m_2 \in M^1 : m_1 \neq m_2. \tag{3.40}$$

Definice 3.41. Množinu všech zpráv M^i tvoří uspořádané trojice $\langle \sigma^i, in^i, out^i \rangle$

$$M^i = \langle \sigma^i, in^i, out^i \rangle = m^i, \tag{3.42}$$

σ^i je symbol in^i, out^i jsou data.

Definice 3.43. Každá zpráva má svého příjemce a odesílatele.

$$\forall P^i : \bigcup_i -M^i = \bigcup_i +M^i. \tag{3.44}$$

Věta 3.44.1. Definovaná přechodová funkce $g^i : S^i \times +M^i \rightarrow -M^i$, jejíž sémantika je následující: P^i je příjemce (receiver) zprávy a P^j je odesílatel (sender) zprávy. Dále platí, že $m^{ij} \in +M^i \wedge m^{ij} \in -M^j$.

Věta 3.44.2. Pro příjemce zprávy pak jsou definovány funkce $data(P^i)$ a $in(m^i)$, pro které platí:

$data(P^i); in(m^i) = in^i; m^i = \langle \sigma^i, in^i, out^i \rangle$.

$in(m^i) = in\langle \sigma^i, in^i, out^i \rangle = m^i$.

Věta 3.44.3. Analogicky jsou definovány funkce $data(P^j)$ a $out(m^i)$ také pro odesílatele zprávy: $data(P^j); out(m^i) = out^i; m^i = \langle \sigma^i, in^i, out^i \rangle$.

$out(m^i) = out\langle \sigma^i, in^i, out^i \rangle = m^i$.

Přestože výměna zpráv probíhá z pohledu sémantiky metody BORM v jednom okamžiku, je-li aplikována teorie konečných automatů, je nutné rozlišovat stav před realizací přechodu a nový stav po realizaci přechodu. Přijatá nebo odeslaná zpráva v čase t+1 bude závislá na stavu v čase t.

Věta 3.44.4. Vzájemnou výměnu dat mezi participanty pak definujeme: pro příjemce jako:
$data^{t+1}(P^i) = data^t(P^i) \cup in(m^{ij}) \wedge in(m^{ij}) \subseteq data^t(P^j)$.

pro odesílatele jako:
$data^{t+1}(P^i) = data^t(P^j) \cup out(m^{ij}) \wedge in(m^{ij}) \subseteq data^t(P^i)$.

Pomocí formálního popisu vycházejícího z teorie konečných automatů byl definován účastník procesu, který si může s ostatními participanty zasílat zprávy. Procesní diagram je tvořen množinou participantů, která je získána složením dílčích konečných automatů do jednoho komplexního automatu. Takový komplexní automat FSM^{BP} pak bude reprezentovat konkrétní procesní diagram. Možnost skládání automatů, čímž dochází k redukci složitosti výsledného procesu, je známa již desítky let (Gill, 1962). Podrobněji se problematikou kompozice, minimalizace a generalizace konečných automatů zabývá například Holzmann (1990) nebo Hopcroft et al. (2001), kteří popisují konkrétní algoritmy pro jejich skládání. Na základě odvozené definice vzájemně komunikující participantů je možné libovolný procesní diagram v metodě BORM popsat jako konečný automat. Tento komponovaný automat bude tvořen množinou konečných automatů, z nichž každý bude reprezentovat právě jednoho participanta. Je-li aplikován na množinu participantů algoritmus pro kompozici konečných automatů popisovaný v Hopcroft et al. (2001), získáme $FSM^{BP} = \langle \Sigma, \Phi, \Omega, \phi, \gamma, \sigma_1 \rangle = P^i = \langle S^i, -M^i, +M^i, f^i, g^i, s_1^i \rangle$,

$$\begin{aligned} \Sigma &= \Pi_i S^j \\ \Phi &= \cup_i - M^i \\ \Omega &= \cup_i + M^i \\ \phi &= \cup_i f^i \\ \gamma &= \cup_i g^i \\ \sigma_1 &= first(\Pi_i s_1^i) \end{aligned} \tag{3.45}$$

Jak je patrné z definice 3.45 na straně 61 pro komponovaný automat, v případě stavů se jedná o kartézské součiny množin stavů jednotlivých participantů. Výsledné množiny odchozích a příchozích zpráv jsou pak sjednocením množiny odchozích, respektive příchozích zpráv jednotlivých participantů. Obdobným způsobem byla odvozena přechodová a výstupní funkce tohoto komponovaného automatu.

Přechodovou funkci pro tento komponovaný automat získáme pomocí přechodových

funkcí f^i dílčích automatů.

$$f : \Sigma \times \Omega \to \Sigma \tag{3.46}$$

$$\begin{aligned} f(Q^t, X) = Q^{t+1};\ (\exists P^i \in BP : S_1, S_2 \in \textstyle\sum(P^i)) \wedge \\ (\exists((S_1 \in Q^t) \wedge (X \in +M^i(S_1))) \wedge \\ \exists(S_2 \in Q^{t+1}) \wedge \\ S_2 = f^i(S_1, X)) \end{aligned} \tag{3.47}$$

Výstupní funkci pro tento komponovaný automat získáme obdobným způsobem pomocí jednotlivých výstupních funkcí g^i dílčích automatů (participantů).

$$g : \Sigma \times \Omega \to \Phi \tag{3.48}$$

$$\begin{aligned} f(Q, X) = X^{'};\ (\exists P^i \in BP : S \in \textstyle\sum(P^i)) \wedge \\ (\exists((S \in Q) \wedge (X \in +M^i(S))) \wedge \\ \exists(X^{'} \in -M^i(S)) \wedge \\ X^{'} = g^i(S, X)) \quad \blacksquare \end{aligned} \tag{3.49}$$

Pomocí výše uvedeného aparátu bylo v práci Moravec (2014) dokázáno, že lze popsat jakýkoli procesní model v BORM pomocí konečného automatu. Praktickým dopadem je podle Moravec (2014) možnost využití veškerých teoretických předpokladů a postupů, které jsou známy z oblasti teorie konečných automatů pro procesní modely v BORM. Lze tedy ověřit, zda jsou všechny stavy účastníků dosažitelné a zda jsou všechny aktivity prováděny. Dále je možné automatizovaně identifikovat stavy, ve kterých by mohlo dojít k uváznutí procesu a tím ověřit konzistenci modelu. Formální popis také otevírá možnosti lepší implementace metody BORM v CASE nástrojích, především v konstrukci simulátoru procesů.

3.3 Strojově čitelný zápis modelu

3.3.1 Domain model

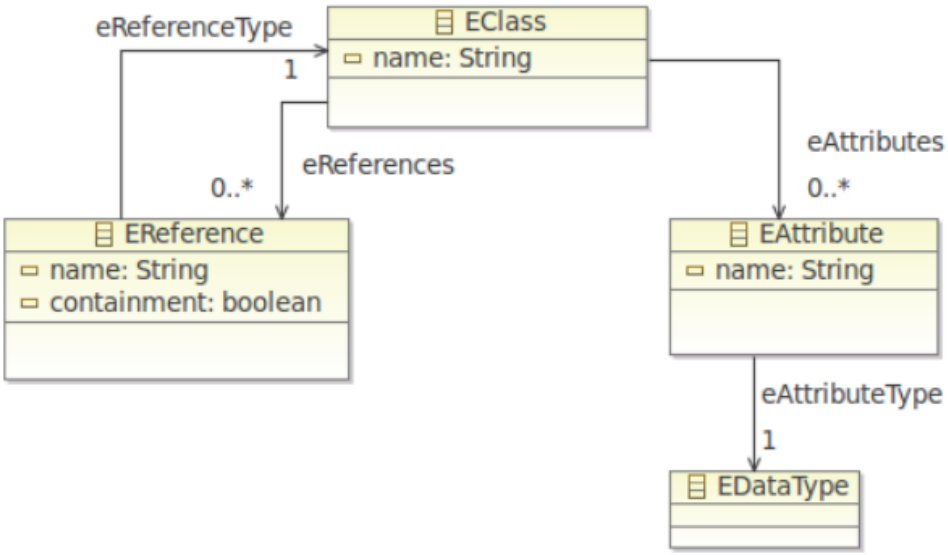
Tůma (2011): Doménový model je v kontextu řešení problémů a softwarového inženýrství konceptuální model domény zájmu. Zápis doménového modelu v diagramu je rozšíření class diagramu dle UML, obsahuje entity, atributy a relace. Nedílnou součástí doménového modelu jsou omezení, které zaručují integritu elementů modelu a podrobněji specifikují elementy doménového modelu. Doménový model je možné definovat pomocí ecore diagramu dle specifikace EMF.

3.3.2 EMF

Tůma (2011): EMF je eclipse modeling framework. Definice z knihy autorů Steinberg et al. (2008): eclipse modeling framework (EMF) je založen na eclipse modelovacím frameworku a zařízení generování kódu pro vytváření nástrojů a dalších aplikací založené na strukturovaném datovém modelu. Z modelové specifikace popsané v XMI, EMF poskytují nástroje a runtime podporu pro vytváření množin Java tříd pro model, množinu zprostředkovávajících tříd, které umožňují zobrazení a na příkazech založenou úpravu modelu a základní editor. Model může být specifikován použitím anotované Java , UML , XML dokumentu, nebo modelovacího nástroje, potom importovaného do EMF. Nejdůležitější ze všeho je, že EMF poskytuje základ pro spolupráci s ostatními EMF založenými nástroji a aplikacemi.

Model v EMF není na tak abstraktní úrovni jako UML. EMF je framework a zařízení pro generování kódu, které umožnuje definovat model v kterékoli z těchto forem. Popis, jak převést a jaké jsou rozdíly mezi class diagramem a doménovým modelem, popisují autoři Steinberg et al. (2008).

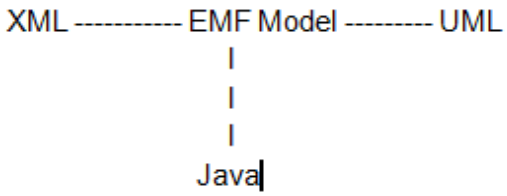
Ecore model Diagram používaný k reprezentaci modelů v EMF se nazývá Ecore. Ecore je sám sobě EMF model a tudíž je svůj vlastní metamodel (Steinberg et al., 2008). Pro ukázku je na obrázku 3.6 na straně 64 zobrazen zjednodušená podmnožina Ecore metamodelu.



Obrázek 3.6: Ukázka zjednodušené podmnožiny Ecore metamodelu, (Steinberg et al., 2008)

3.3.3 Standardy modelování spojené s EMF

Tůma (2011): EMF souvisí s důležitými modelovacími standardy od OMG (Object Modeling Group) jako UML, MOF, MDA a XMI. EMF je zaměřeno pouze na jedenu z možností UML (Unified Modeling Language) modelování, a to modelování tříd. Meta object facility se soustřeďuje na popis tříd v UML ve spojení s uložištěm dat. Lze říct, že ekvivalentem pro MOF (Meta-Object Facility) v UML je v EMF Ecore. Návaznost mezi spojitostmi s EMF jsou vidět na obrázku 3.7 na straně 64 (Steinberg et al., 2008).



Obrázek 3.7: Propojení s EMF, od autora (Steinberg et al., 2008)

Tůma (2011): XMI (XML Metadata Interchange) je standard, který spojuje modelování s XML. XMI definuje jednoduchou cestu, jak serializovat modely v XML dokumentu. Struktura XMI dokumentu je úzce spojena s odpovídajícím modelem se stejnými názvy a hierarchií elementů, která sleduje hierarchii obsaženou

v modelu. Vztah mezi modelem a jeho XMI serializací je snadno pochopitelný. XMI je používáno jako serializační formát pro instance kteréhokoli EMF modelu a je vhodný pro použití s modely reprezentující metadata (např. Ecore) (Steinberg et al., 2008).

MDA (Model Driven Architecture) je průmyslová architektura navržená OMG. MDA adresuje celý životní cyklus vývoje aplikací, dat a standardy integrace aplikací, které pracují s vícenásobnými jazyky middleware a výměnnými formáty. MDA unifikuje některé nejlepší průmyslové techniky v softwarové architektuře, modelování, management metadat a softwarové technologie, které umožňují uživateli vyvíjet modelové specifikace jednou a zacílit vícenásobně technologické implementace použitím transformací a mapování. EMF podporuje klíčový MDA koncept použití modelů jako vstupu k vývojové a integrační nástroje. V EMF je model použit k řízení generování kódu a serializaci při výměně dat (Steinberg et al., 2008).

3.4 Transformace modelů

3.4.1 Model-Driven Architecture

Model-Driven Architecture (MDA) definován v roce 2003 Architecture (2003),Frankel (2003) je iniciativa Object Management Group (OMG) a definuje přístup softwarového vývoje programů založeného na modelování a automatickém převodu modelů na implementaci. Základní MDA šablona definuje platformě nezávislý model (PIM - Platform Independent Model) a jeho automatický převod na jeden nebo více platformě specifických modelů (PSMs - Platform Specific Models) .

Přístup MDA přináší výhody například přenosnost navzdory oddělení znalostí aplikace od převodu na konkrétní implementační technologii, zvýšení produktivity díky automatickému převodu, zvýšení kvality díky znovupoužitelnosti pomocí již prověřených šablon nejlepších praktik v převodu a zvýšení konzistence a sledovatelnosti mezi modelem a kódem.

Současné OMG standardy jako například Meta Object Facility (MOF) od tvůrců z roku 2003 OMG (2003*b*) a UML od tvůrců z roku 2001 OMG (2001) poskytují

Tabulka 3.4: Přehled existujících přístupů v transformacích modelů, (Czarnecki & Helsen, 2006)

Nástroj	Komentář
GreAT	GreAT byl v roce 2003 na stránkách (Agrawal et al., 2003).
UMLX	UMLX je v publikaci od autora Willink (Willink, 2003).
ATOM	VIATRA byla v roce 2002 na stránkách (Varró et al., 2002).
VIATRA	FUUT-je and GMT o tomto nástroji více na FUUT-je (FUUT-je, 2005).
BOTL	BOTL o tomto přístupu píší Braun a Marshall (P. Braun, 2003), (Marschall & Braun, 2003).
ATL	ATL přístup popisuje kolektiv autorů v čele s Bezivin. (Bézivin et al., 2003).
	dále lze nalézt přístupy v publikacích :
QVT-Partners	v publikaci od autorů QVT-Partners (QVT-Partners, 2003).
CBOP, DSTC a IBM	oproti tomu publikace od autorů CBOP, DSTC a IBM (CBOP & IBM, 2003).
Alcatel	následuje publikace od autora Alcatel a dalších (Alcatel et al., 2003).
Interactive Objects and Project Technology	dále od Interactive Objects and Project Technology (OMG, 2003*a*).
CC	závěrem přístup od CC (CC, 2007).

velmi dobrou podporu pro definování PIM a PSM.

Porovnáním přístupů transformací modelů a přehledem existujících přístupů je možné vidět v tabulce 3.4 na straně 66.

- Nástroje s otevřeným zdrojovým kódem (open source) jsou uvedeny v tabulce 3.5 na stráně 67.
- Komerční nástroje a jejich přehled je k vidění v tabulce 3.6 na straně 67.

3.4.2 Kategorie přístupů transformací

Na nejvyšší úrovni lze rozlišit přístupy transformací modelu na kód a modelu na model. Obecně lze říci, že transformace modelu na kód je speciální případ trans-

Tabulka 3.5: Nástroje s otevřeným zdrojovým kódem, (Czarnecki & Helsen, 2006)

Nástroj	Komentář
Jamda	Jamda byla k nalezení v odkazu z roku 2003 (Jamda, 2003).
AndroMDA	AndroMDA, o které je možné nalézt více informací na AndroMDA (AndroMDA, 2003).
JET	O JET je více informací k nalezení na JET z roku 2002 (JET, 2002).
FUUT-je and GMT	FUUT-je and GMT o tomto nástroji více na FUUT-je (FUUT-je, 2005).

Tabulka 3.6: Komerční nástroje, (Czarnecki & Helsen, 2006)

Nástroj	Komentář
OptimalJ	OptimalJ je k dopátrání v roce 2006 OptimalJ (OptimalJ, 2006).
ArcStyler	ArcStyler byl v roce 2004 na odkazu ArcStyler (ArcStyler, 2004).
XDE	XDE je novější než ArcStyler je XDE z roku 2005 (XDE, 2005).
b+m Generator Framework	b+m Generator Framework je v odkazu z roku 2006 na b+m ArchitectureWare (b+m ArchitectureWare, 2006).

formace modelu na model, pouze je potřeba poskytnout metamodel pro cílový programovací jazyk. Z praktických důvodů využití existujících kompilátorů je kód generován pouze jako text, na který je použit kompilátor. Z tohoto důvodu jsou rozlišovány transformace modelu na kód a modelu na model. Několik nástrojů např. Jamda, XDE, a OptimalJ nabízí jak transformaci modelu na model, tak také modelu na kód (Czarnecki & Helsen, 2003).

V kategorii modelu na kód je rozlišováno mezi přístupem založeným na visitor a na šabloně. V kategorii modelu na model jsou rozlišovány přístupy přímé manipulace, vztahy, transformace přes graf, hybridní přístupy řízené struktury (Czarnecki & Helsen, 2003).

3.4.3 Transformace modelu na text

Přístup založený na visitor

Základní přístup generování kódu spočívá v poskytnutí mechanismu visitor k projití vnitřní reprezentace modelu a psaní kódu do textového proudu. Příkladem tohoto

přístupu je Jamda, která je objektově orientovaný framework poskytující třídy k reprezentaci UML modelů, API pro manipulaci s modely a mechanismus visitor (nazývaný také CodeWriters) ke generování kódu. Jamda nepodporuje MOF standard k definování nového metamodelu, nicméně nový typ metamodelu může být uveden pomocí podtříd existujících Java tříd, které reprezentují předdefinované typy elementů modelu (Czarnecki & Helsen, 2006).

Přístup založený na šablonách

Většina současných dostupných MDA nástrojů podporují generování kódu z modelu založeném na šablonách, například:

- b+m Generator framework,
- JET,
- FUUT-je,
- Codan Architect,
- AndroMDA,
- ArcStyler,
- OptimalJ a XDE (OptimalJ a XDE také poskytují transformace modelu na model).

AndroMDA používá open-source technologii generující kód založenou na šablonách, která se jmenuje Velocity k nalezení v odkazu (XDoclet, 2006), (Velocity, 2003).

Šablona se běžně sestává z cílového textu obsahující části metakódu k přístupu k informacím ze zdroje a vykonává výběr kódu a postupné rozšíření (Cleaveland, 2001*a*). Vzhledem k terminologii LHS (levostranné - Left Hand Side), která používá spustitelnou logiku k přístupu ke zdrojům a RHS (prostranné - Right Hand Side), která kombinuje textové řetězce se spustitelnou logikou pro výběr kódu a postupné rozšiřování, není žádné syntaktické oddělení mezi LHS a RHS. Přístup pomocí šablon běžně nabízí uživatelsky definované časování ve vnitřní formě volání šablon z jiné šablony (Czarnecki & Helsen, 2006).

Logika přístupu LHS ze zdrojového modelu může mít různé formy. Může být jednoduše kód v Java, na který se přistupuje pomocí poskytnutého API vnitřní reprezentace zdrojového modelu (například JMI), nebo může být logika vytvořena pomocí dotazování (například OCL nebo XPath od tvůrců z W3C z let 1999 (W3C, 1999)). Framework b+m Generator propaguje myšlenku oddělené více komplexní zdrojové přístupové logiky (která může potřebovat a získávat informace z různých míst zdrojového modelu) z šablon přesunutím jejich do uživatelsky definovaných operací z elementů zdrojového modelu (Czarnecki & Helsen, 2006).

V porovnání s transformací založenou na visitor, struktura šablon se více podobá kódu, který je generován. Šablony dovolují, aby mohly být postupně vyvíjeny, tak jak mohou být odvozeny z příkladu. Přístup pomocí šablon využívá textu. Šablony mohou obsahovat syntakticky nebo sémanticky nesprávné fragmenty kódu. Na druhé straně, textové šablony jsou nezávislé na cílové platformě/jazyku a usnadňují generování jakéhokoli textového artefaktu i dokumentace. Podobná technologie je rámcové zpracování, které rozšiřuje šablony pomocí více sofistikovaného mechanismu přizpůsobení a adaptace (rámce) (Czarnecki & Helsen, 2006):

- Basset byl aktuální v roce 1997 (Bassett, 1997).
- XFramer je z roku 2003 (Emrich, 2003).
- ANGIE je od Delta Software Technology (Delta, 2004)).
- FPL je z roku 2004 (FPL, 2004).
- XVCL oproti předešlému je z roku 2005 (FXVCL, 2005).

3.4.4 Transformace modelu na model

Transformace modelu na model převádí data mezi zdrojovým a cílovým modelem. Ten může být instancí stejného nebo odlišného metamodelu. Všechny tyto přístupy podporují syntaktické psaní proměnných a šablon.

Většina existujících MDA nástrojů poskytuje pouze transformace modelu na kód, který používají na generování PSMs z PIMs. Důvod, proč je potřeba transformace modelu na model je ten, že přemosťují abstraktní mezery mezi PIMs a PSMs, a tím je snazší transformovat data mezi modely, než přímo do cílového PSM. Například

mezi diagramem tříd a EJB implementací lze pomocí nástroje OptimalJ generovat komponentní model mezi EJB, který obsahuje všechny potřebné informace k vytvoření Java kódu z něj (Czarnecki & Helsen, 2003).

Přístup přímé manipulace

Tento přístup nabízí reprezentaci vnitřního modelu a k tomu API pro ovládání. Nástroje s tímto přístupem jsou běžně implementovány jako objektově-orientované frameworky, které mohou poskytnout minimální prostředí k organizaci transformací (Czarnecki & Helsen, 2003).

Uživatelé musí naimplementovat pravidla transformace a rozvržení většinou od začátku s použitím programovacího jazyku, jako je například Java. Příkladem tohoto přístupu je nástroj Jamda. Ten implementuje transformace přímo, narozdíl některé MOF-příbuzné API, například JMI (Czarnecki & Helsen, 2003).

Relační přístup

V této kategorii deklarativního přístupu, kde se jako hlavní koncept využívá matematických vztahů Akehurst et al. (2003), QVT-Partners (2003), CBOP & IBM (2003), Gerber et al. (2002) a převodních pravidel v publikaci (Alcatel et al., 2003), je základní myšlenkou zápis zdrojového a cílového elementu typu vztahu a specifikace omezení jeho použití. V této čisté formě pouze specifikace není spustitelná Akehurst & Kent (2002), vztahy publikovali QVT-Partners (2003) a převodní pravidla Alcatel et al. (2003).

Deklarativním omezením může být přiřazena spustitelná sémantika, jako v logickém programování. Ve skutečnosti logické programování s jeho jednotně založenou shodou, vyhledáváním a zpětnou cestou, může být využito k popisu vztahů. Gerber et al. (2002) prozkoumali použití logického programování (částečně Mercury a typový dialekt Prologu a F-logiky objektově orientovaného logického paradigmatu) k implementaci transformací. Návrh QVT autory CBOP & IBM (2003) byl inspirován přístupem F-logiky. QVT-Partners (2003) rozděluje dva vztahy, které jsou v jejich frameworku, nespustitelnou a obousměrnou specifikaci transformace a transformaci samotnou, která je spustitelná, jednosměrná a implementuje vztahy

Tabulka 3.7: Příklady přístupu grafové transformace, (Czarnecki & Helsen, 2006)

Přístup
VIATRA
ATOM
GreAT
UMLX
BOTL

v transformaci.

Všechny přístupy pomocí vztahů jsou bez vedlejších účinků. Často je podporována zpětná cesta, jako jsou například uvedeny v publikaci (Gerber et al., 2002) a (QVT-Partners, 2003), na rozdíl s imperativním přístupem přímé manipulace vytvářejí se elementy implicitně (Gerber et al., 2002) a (CBOP & IBM, 2003). Vztahovou specifikaci, uvádí Akehurst & Kent (2002), vztahy QVT-Partners (2003) a převodní pravidla Alcatel et al. (2003), mohou být vykládány obousměrně. Přístup založený na logickém programování také přirozeně podporuje obousměrnost. Některé přístupy ale upravují směr pro vykonatelnost transformací, jako například přístup, který uvádí CBOP & IBM (2003) a převod, který uvádí QVT-Partners (2003). O přístupu programování založený na logice píše (Gerber et al., 2002) a (CBOP & IBM, 2003). Ten vyžaduje striktní oddělení mezi zdrojovým a cílovým modelem tzn., že nedovoluje vmístěnou aktualizaci (Czarnecki & Helsen, 2003).

Přístupy transformací založených na teorii grafů

Tato kategorie přístupu transformací modelu se zakládá na teorii transformaci grafů. Tento přístup pracuje s typovým, atributovým a popisným grafovým základem (Akehurst & Kent, 2002). Jde o druh grafu speciálně navržený k návrhu reprezentace UML a podobných modelů. Příklady přístupu grafové transformace na transformaci modelů obsahují přístupy uvedené v tabulce 3.7 na straně 71.

Pravidla transformace grafů obsahují LHS grafové šablony a RHS grafové šablony (Czarnecki & Helsen, 2006).

Grafové šablony mohou být výstupem v konkrétní syntaxi jejich zdrojového nebo cílového jazyka, jako je například ve VIATRA, nebo jako je v MOF abstraktní syntaxi (například BOTL). Výhodou konkrétní syntaxe je skutečnost, že jsou s ní lépe seznámeni vývojáři pracující s modelovacím jazykem, než se syntaxí abs-

traktní. UML šablony v konkrétní syntaxi vedou ke zestručnění, narozdíl od šablon v odpovídající abstraktní syntaxi (Braun & Marschall, 2003), (Andries et al., 1999).

Na druhé straně je jednodušší poskytnout obecný výstup pro abstraktní syntaxi, která bude pracovat s jakýmkoli metamodelem, který je použitelný, když není dostupný žádný jiný speciální syntaktický metamodel.

Šablona LHS je v modelu rozpoznána, transformována a je nahrazena šablonou RHS. Šablona LHS často obsahuje podmínky (například negativní podmínky). Některá přidaná logika (například doména textových řetězců a číselných hodnot) je potřeba v závislosti na výpočtu cílových atributů hodnot jako jména elementů. GreAT nabízí rozšířenou formu šablon s multiplikací na hranách a uzlech. Většina přístupů rozvrhování má externí formu a rozvrhovací mechanismus obsahující nedeterministický výběr, explicitní podmínky a pevné iterace. Pevné iterace jsou částečně použitelné pro výpočet tranzitivních uzávěrů.

K relačním přístupům jsou podobné přístupy grafové transformace. Ty jsou vyjádřením modelové transformace v deklarativním smyslu (Czarnecki & Helsen, 2006).

Přístup orientovaný strukturou

Tento přístup má dvě odlišné fáze. První fáze je zaměřena na vytvoření hierarchické struktury cílového modelu, druhá fáze je nastavení atributů a reference na cílový model. Celý framework rozhoduje rozvrhování a aplikační strategii. Uživatelé jsou zaměření pouze na poskytnutí transformačních pravidel.

Příkladem přístupu orientovaného strukturou je transformace modelu na model ve frameworku poskytnutém OptimalJ. Tento framework je implementován v Java a poskytuje uživatelům podtřídy k definování transformačních pravidel. Základem je kopírování elementů ze vstupního modelu do výstupního modelu. Elementy mohou být přizpůsobeny k dosažení požadovaného efektu transformace. Framework používá reflexe k poskytnutí deklaračního rozhraní. Transformační pravidla jsou implementovány jako metoda s vstupními parametry. Typ parametru rozhoduje zdrojový typ pravidla a metoda vracející objekt Java, který reprezentuje třídu cílového elementu modelu. Pro pravidla není povoleno mít vedlejší efekt a rozvrhování je celé pod frameworkem (Czarnecki & Helsen, 2003).

Jiný přístup řízený strukturou je znázorněn v publikaci OMG (2003*a*). Speciální vlastnost tohoto přístupu je na cíl orientovaná organizace pravidel, kde je jedno pravidlo na cílový typ elementu a zanořená pravidla odpovídající obsahovou hierarchii v cílovém metamodelu. Vykonání tohoto modelu může být jako konfigurace shora dolu cílového modelu.

Přístupy hybridní

Hybridní přístupy kombinují odlišné techniky z předešlých kategorií. Jazyk transformačních pravidel (Transformation Rule Language - TRL), jak popisuje Alcatel et al. (2003) je složením deklarativního a imperativního přístupu. TRL také může být klasifikován v relační kategorii, ale rozhodnutí o zařazení byl ponechán samostatně, protože má silné imperativní komponenty. Podobně uvádí QVT-Partners (2003), který rozlišujeme na specifikaci a implementaci. Mapovací pravidlo v TRL deklaruje vztah mezi zdrojovými a cílovými elementy, který je omezený množinou invariantů. TRL jsou podobné vztahům, které uvádí QVT-Partners (2003) a zapadají do relační kategorie.

Operační pravidla v TRL představují spustitelná transformační pravidla. Oproti převod-ním pravidlům, operační pravidla mají explicitní stav. Explicitní stav znamená, zda pravidlo vytváří, aktualizuje nebo maže elementy. Rozvrhování je explicitní ve vnitřní formě, kde pravidla explicitně volají ostatní pravidla ve svém těle. Dědičnost pravidel je podporována. Pravidla mohou být organizována do modulů, které se nazývají jednotky. Dědičnost mezi moduly s přepisem je také podporována (Czarnecki & Helsen, 2006).

Transformační jazyk Atlas (Atlas Transformation Language - ATL), jej popisuje Bézivin et al. (2003), je také hybridní přístup, který má některé podobnosti s TLR. Transformační pravidlo v ATL může být plně deklarativní, hybridní nebo plně imperativní. Šablona LHS plně deklarativního pravidla, které může být zdrojovou šablonou, se skládá z množiny syntaktických typových proměnných s volitelným OCL omezením jako filtr nebo navigační logika. RHS plně deklarativního pravidla, které se také nazývá cílová šablona, obsahuje množinu proměnných a některou deklarativní logiku k navázání hodnoty atributů v cílových elementech. V hybridních pravidlech jsou zdrojová a cílová šablona nahrazovány blokem imperativní logiky, která je spuštěna po aplikování cílové šablony. Pravidla jsou obousměrná a pod-

porují dědičnost pravidel. ATL striktně odděluje zdrojový a cílový model, transformace mohou být díky automatickému mechanismu kopírovány. ATL poskytuje jak implicitní, tak i explicitní rozvrhování. Algoritmus implicitního rozvrhování začíná voláním pravidla, které bylo navrženo jako vstupní bod, a pak volá další pravidla. Po dokončení první fáze automaticky kontroluje shody zdrojových šablon a vykonává odpovídající pravidla. Nakonec je vykonán konečný bod. Explicitně interní rozvrhování je podporováno možností volat pravidla z imperativního bloku jiného pravidla (Czarnecki & Helsen, 2006).

XDE je příkladem vysoce hybridního přístupu. XDE podporuje transformaci modelu na model přes mechanismus šablon. Originální motivací pro šablony v XDE bylo poskytnutí automatizované aplikace návrhových vzorů Gang of Four, jak popisuje ve své publikaci Helm et al. (2002). Základním konceptem mechanismu šablon v XDE je parametrizovaná spolupráce, která je UML mechanismem v návrhu modelovacích šablon. Obecně je transformace modelu na model podcíl, základní mechanismus šablon je vyvinut ve vysoce hybridní a komplexní přístup (XDE, 2005).

Kapitola 4

Analytická část

4.1 Metoda automatizované transformace modelu na text

Tato kapitola je zaměřena na dokumentaci transformace modelu na text resp. modelu, který je reprezentován notací BORM (Business Object Relation Modeling) a textem, který je reprezentován reportem založeném na HTML. Součástí návrhu metody transformace model na model bylo rozšíření nástroje OpenCASE, které podporuje modelování v notaci BORM. Toto rozšíření podporuje generování reportu, který je srozumitelný pro obchodní analytiky. Report je založený na HTML. HTML bylo zvoleno z důvodu přenositelnosti a rozšířenosti. Základ pro tento generovaný report je ale širší a nabízí další formáty, jako jsou txt, tex a pdf. Samotná transformace je představena na případové studii. Transformace je z modelu úrovně řízení na model úrovně provozní. Provozní úroveň modelu je textová a zaměřená na každého participanta.

4.1.1 Úvod

Model je v CASE nástroji nejvhodnějším dokumentačním systémem, ale v reálných situacích není potřeba komplexní model, jelikož není pohodlné ani vhodné pracovat s komplexním modelem. Toto je důvod proč je zapotřebí nástroj, který umožňuje generovat dokumentaci každého aspektu modelu. Zdrojem pro gene-

rování dokumentace je repositář CASE nástroje. Když je dokumentace automaticky generována z modelu, je zaručena konzistence mezi modelem a dokumentací. Další výhodou nástroje je možnost znovu vygenerování dokumentace, pokud dojde ke změně modelu. Proč je tento výstup potřebný pro např. obchodní analytiky? Důvodů je více, ale obecně lze říci, že dochází k hlubšímu porozumění modelované situace, než když je pouze popsána diagramem na obrázku. BORM diagram je velmi dobře srozumitelný, ale ne pro všechny stejně (např. obchodní analytiky). To je důvod, proč HTML report napomáhá k hlubšímu porozumění širší skupiny uživatelů (např. obchodní analytici). HTML report přináší snadnější dostupnost k informacím popsaným diagramem BORM. V této kapitole je prezentován přístup flexibilního modelování obchodních procesů na obou vrstvách, jak na řídící, tak také na provozní vrstvě. Tento přístup je složen z kombinace metody modelování BORM - Business Object Relation Modelling a vývoje původního softwarového nástroje OpenCASE, který provádí automatizované transformace modelů.

Tato kapitola je zaměřena na transformaci zdrojového modelu na výstupní model a také na použité technologie při implementaci této transformace. Vstupem do této transformace je model používající notaci BORM. Výstup z této transformace je model, který je reprezentován dokumentací - HTML reportem.

Cílem reportu v HTML je dosažení podobné dokumentace (SBVR, 2008) a tím překlenutí mezery mezi obchodními analytiky a návrháři informačních systémů (IS) (Cabot et al., 2010).

Celý tento proces je založen na metodě BORM, která byla popsána v mnoha publikacích (Knott et al., 2000), (Knott et al., 2006), (Struska & Merunka, 2007), (Struska & Pergl, 2009).

Metoda BORM a notace BORM ORD je použita jako vstup transformace používané v nástroji OpenCase (Pergl & Tůma, 2012*a*). Jádro transformace je založeno na přístupu HOT (High Order Transformation) (Brambilla et al., 2009).

Tato transformace je prezentována na případové studii, která demonstruje transformaci modelu na model. Vstupním modelem je obchodně procesní model úrovně obchodně řídící a výstupem je procení model úrovně obchodně provozní. Konkrétně se případová studie zabývá procesem z vyšetřovací činnosti (kriminalistiky) a to identifikací pachových vzorků (MPI - Metoda Pachové Identifikace).

4.1.2 Formální popis transformace

4.1.3 Zobrazení BORM ORD a Report

Na levé straně tabulky 4.1 na straně 77 jsou zobrazeny prvky z notace BORM ORD a na pravé straně jsou elementy z dokumentace (Reportu). Bližší informace o BORM ORD lze nalézt u autora Knott et al. (2000), Knott et al. (2003).

Participant	$\Leftrightarrow$	Tabulka
Začátek role	$\Leftrightarrow$	První řádek v tabulce
Konec role	$\Leftrightarrow$	Konečný řádek v tabulce
Aktivita, Stav	$\Leftrightarrow$	Řádek v tabulce - paragraf
Přechod	$\Leftrightarrow$	Nový řádek v tabulce
Datový tok	$\Leftrightarrow$	Odkaz na řádek
Podmínka	$\Leftrightarrow$	Nový sloupec v tabulce

Tabulka 4.1: Tabulka zobrazení BORM ORD a reportu zpracováno autorem

4.1.4 Formální popis zobrazení

Definice 4.1. Textový report je reprezentován modelem, který lze vyjádřit jako složení tabulek T_i reprezentující jedinečnou modelovanou entitu a definován jako uspořádaná pětice

$$T_i(R_i, I_i, \delta_i, pr_{i0}, kr_{ie}), \text{ kde:} \tag{4.2}$$

- R_i je konečná neprázdná množina všech stavů, v nichž se participant může nacházet.
- I_i je konečná neprázdná množina všech vstupů.
- δ_i reprezentuje vykonávané aktivity, tedy přechody mezi stavy; $\delta_i : I_i \times R_i \Rightarrow R_i$.
- pr_i je počáteční stav procesu a platí, že $pr_{i0} \in R_i$.
- kr_i je koncový stav procesu a platí, že $kr_{ie} \in R_i$.

Ekvivalence Participanta z notace BORM ORD s tabulkou v dokumentací reportu je patrná v tabulce 4.2 na straně 78. Obě tyto entity modelu lze vyjádřit jako prvky konečného automatu.

Participant		Tabulka
S_i	$\Leftrightarrow$	R_i
I_i	$\Leftrightarrow$	I_i
$\delta_i : I_i x S_i -> S_i$	$\Leftrightarrow$	$\delta_i : I_i x R_i -> R_i$
$s_i; s_i \in S_i$	$\Leftrightarrow$	$pr_i; pr_i \in R_i$
$s_i; s_i \in S_i$	$\Leftrightarrow$	$kr_i; kr_i \in R_i$

Tabulka 4.2: Tabulka ekvivalence popisu participanta a tabulky v reportu zpracováno autorem

4.2 Metoda automatizované transformace modelu na model

V této části představen přístup, který je založen na kombinaci konečného automatu a objektově orientovaného přístupu. Myšlenkou je modelování obchodních procesů a případů jako konečných automatů. tato část se zaměřuje na přístup BPMN, přístup BORM a jejich propojení.

Hlavní částí byla transformace modelu na model založená na BORM. Tato kapitola navazuje na kapitolu 4.1 Metoda automatizované transformace modelu na text na straně 75. Návrh transformace byl mezi BPMN a BORM.

4.2.1 Úvod

Tato část představuje dva systémy a znalostní modelovací techniky, které mohou být použity jako nástroje ke koordinaci a komunikaci mezi zastoupenými stranami obchodními analytiky a návrháři informačních systémů. Kapitola se zaměřuje na obecný přístup BPMN popsaný v kapitole 3.2.1 Business Process Model and Notation na straně 35 a přístup BORM popsaný v kapitole 3.2.2 Business Object Realtion Modeling na straně 42.

Transformace mezi těmito dvěma technikami byla založena na MDA uvedeno v kapitole 3.4.1 Model-Driven Architecture, jak je popsáno na straně 65.

Motivací bylo propojení těchto dvou technik. Na jedné straně populární a rozšířené BPMN a na straně druhé starší, ale ve své době inovativní BORM.

4.2.2 Cíl

Hlavní cíl bylo ukázat vazby a pokračování procesu vývoje založeném na metodě BORM.

Cílem bylo navrhnout automatizovanou transformaci modelů. Modely jsou reprezentovány BORM ORD (Objektově relační diagram) a BPMN PSD (Procesně strukturovaný diagram).

Dílčími cíli byly verifikace a validace metody použité pro transformaci. Verifikace byla vytvořena testováním algoritmu transformace nad případovými studiemi. Validace byla provedena diskuzí, která byla provedena porovnáním s ostatními přístupy BPM transformací, BPM jsou v kapitole 3.1.1 Modelování podnikových procesů popsané na straně 11 .

4.2.3 Metoda

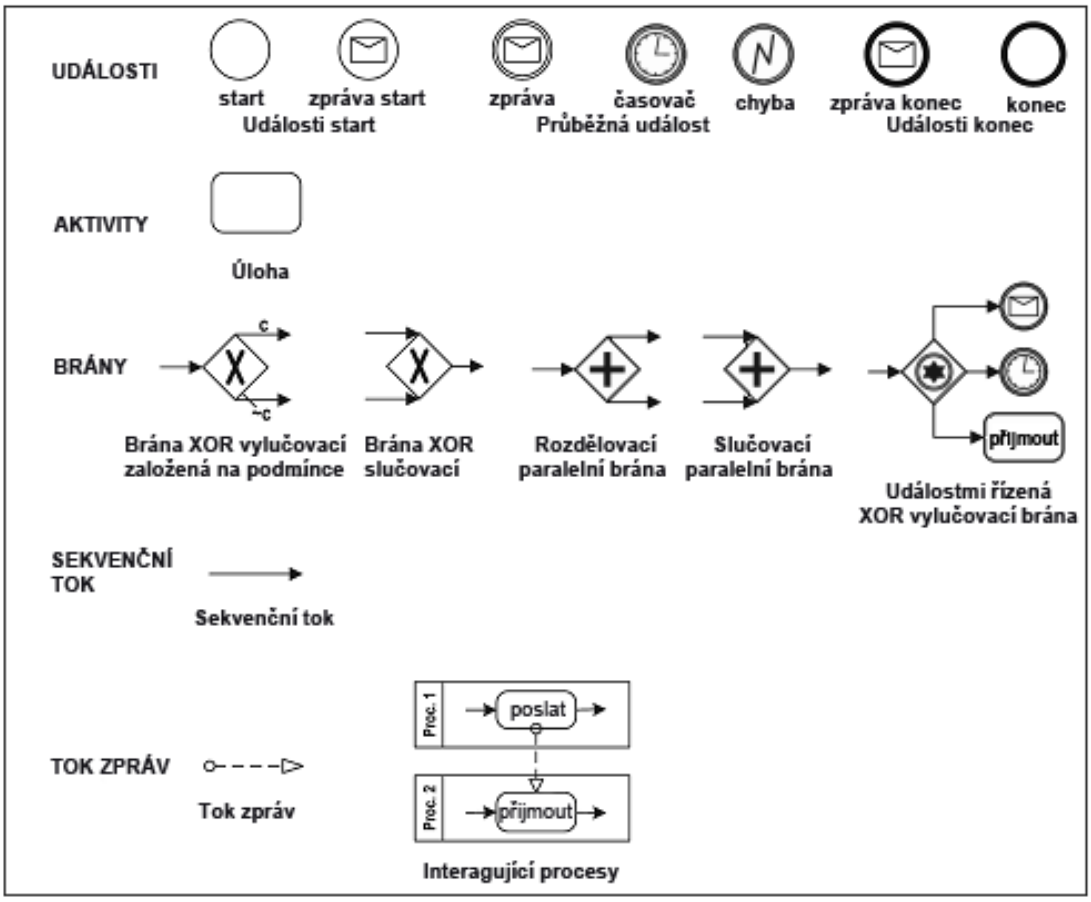
Metoda byla založena na postupných krocích. Prvním krokem byla analýza modelů BPMN PSD popsaný v kapitole 3.2.1 Business Process Model and Notation na straně 35 a přístup BORM ORD popsaný v kapitole 3.2.2 Business Object Realtion Modeling na straně 42. Dalším krokem bylo nalezení spojitostí mezi jednotlivými objekty modelů. Dále byla provedena formální definice spojitostí přes petriho sítě u vedené k kapitole 3.1.3 Petriho sítě na straně 18.

4.2.4 Elementy BPMN a BORM

4.2.5 BPMN

Vybraná množina elementů z BPMN je zobrazena na obrázku 4.1 na straně 80. Tyto elementy byly vybrány vzhledem k publikacím od autorů (Dijkman & Dumas, 2007) a (Lam, 2012). Autor v publikaci (Lam, 2012) uvádí, jak jsou elementy v BPMN redundantní, a ukazuje množinu nepřekrývajících se elementů[1]. Vybrané elementy jsou podmnožinou ze souhrnu elementů v metamodelu BPMN, jak je možné vidět na obrázku 10.22 na straně 155.

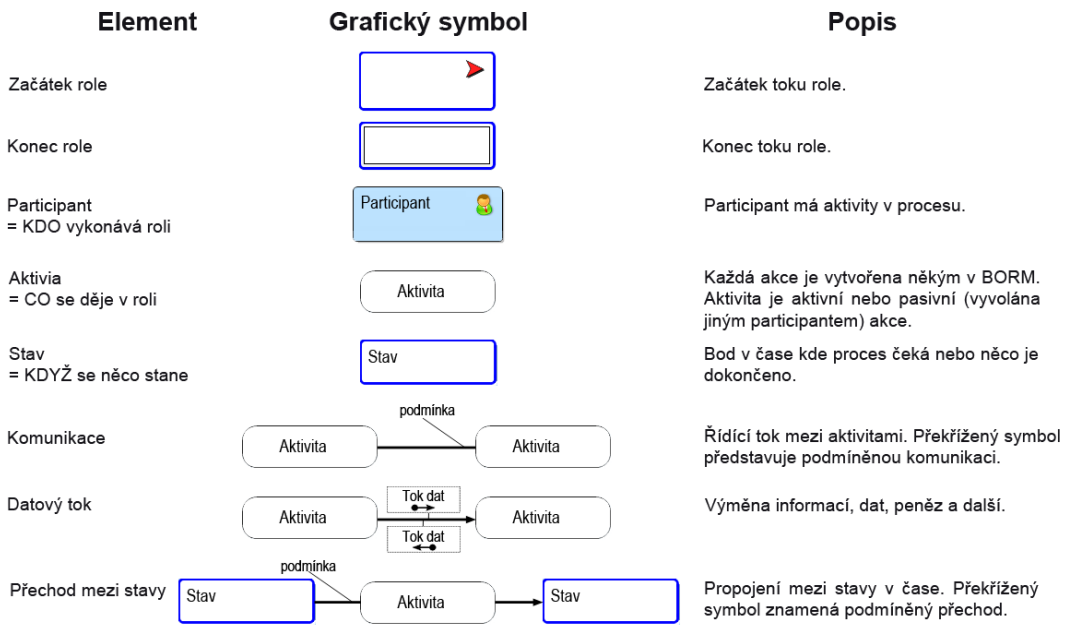
[1]Úprava množiny je rozebrána v kapitole diskuze 6 na straně 108



Obrázek 4.1: Podmnožina BPMN elementů čerpána z (Dijkman & Dumas, 2007) a zpracována autorem

4.2.6 BORM

Vybraná množina elementů z BORM je zobrazena na obrázku 4.2 na straně 81. Tyto elementy byly vybrány vzhledem k spojitosti mezi modely jak je uvedeno v kapitole 4.2.8 Spojitosti mezi elementy BORM a BPMN popsané na straně 83. Vybrané elementy jsou podmnožinou ze souhrnu elementů v metamodelu BORM, jak je možné vidět na obrázku 10.20 na straně 154.



Obrázek 4.2: Podmnožina BORM ORD elementů čerpána z (Molhanec & Merunka, 2011) a zpracována autorem

4.2.7 Komplexní metamodel převedený na metamodel vybraných elementů

PS diagram BPMN

Celý metamodel elementů BPMN je zobrazen na obrázku 10.22 na straně 155. Tento metamodel obsahuje úplný výčet elementů diagramu PS z notace BPMN. Z tohoto metamodelu byl vytvořen metamodel obsahující elementy, které jsou koherentní v závislosti na diagram OR z metody BORM. Metamodel vybraných koherentních elementů diagramu PS z notace BPMN je zobrazen na obrázku 10.21 na straně 155.

Na obrázku 10.21 je vyobrazen metamodel v podobě diagramu tříd. Diagram tříd se skládá ze tříd:

- Process, což v překladu znamená Proces.
- Activity, což v překladu znamená Aktivity.
- Flow, což v překladu znamená Tok.

- Event, což v překladu znamená Událost.
- Task, což v překladu znamená Úloha.
- SequenceFlow, což v překladu znamená Sekvenční tok.
- MessageFlow, což v překladu znamená Tok zpráv.
- End, což v překladu znamená Konec.
- Intermediate, což v překladu znamená Průběžná.
- Start, což v překladu znamená Start.
- Gateway, což v překladu znamená Brána.
- Pool, což v překladu znamená Bazén.

Proces je složen z aktivit. Vztah je proces a může obsahovat až mnoho aktivit, a právě jedna konkrétní aktivita může být obsažena v procesu jednou nebo neobsažena. Proces je rozdělen na bazény, kterých může být až mnoho. Od aktivity dědí úloha.

Tok je předkem sekvenčního toku a toku zpráv. Sekvenční tok propojuje aktivity mezi sebou a brány s aktivitami. Tok zpráv propojuje bazény, a to právě tak, že jeden tok zpráv propojuje dva bazény.

Události jsou propojené příchozími a odchozími toky, které se dělí, jak již bylo zmíněno, na sekvenční toky a toky zpráv. Události se dělí na start, průběžnou a konec, tedy start, průběžná a konec jsou potomky události.

OR diagram BORM

Metamodel na obrázku 10.19 na straně 153 a 10.20 na straně 154 popisuje OR diagram z metody BORM. Na obrázku 10.19 na straně 153 je zobrazen metamodel uvažovaný pro transformaci modelu na model, konkrétně modelu PS diagramu BPMN na model OR diagramu metody BORM. Dále na obrázku 10.20 na straně 154 je vyobrazen metamodel použitý v nástroji OpenCASE, tedy tento metamodel je více zaměřen z realizační fáze. Na obrázku 10.19 na straně 153 je vyobrazen metamodel v podobě diagramu tříd. Diagram tříd se skládá ze tříd:

- Participant, což v překladu znamená Participant.
- Event, což v překladu znamená Událost.
- State, což v překladu znamená Stav.
- Activity, což v překladu znamená Aktivita.
- Transition, což v překladu znamená Přechod mezi stavy.
- Condition, což v překladu znamená Podmínka.
- DataFlow, což v překladu znamená Datový tok.
- Communication, což v překladu znamená Komunikace.
- StateType, což v překladu znamená Typ stavu.
- Start, což v překladu znamená Start.
- Inner, což v překladu znamená Vnitřní.
- Final, což v překladu znamená Konec.

Participant je složen z událostí, které se dále dělí na aktivity a stavy. Formálně stav a aktivita jsou potomky události, tedy stav a aktivita dědí od předka události. Mezi událostmi jsou přechody mezi stavy, které mohou být podmíněné. Podmíněné přechody mezi stavy jsou pojmenované, jméno odpovídá podmínce. Stav má typ, který se dělí podle výčtového typu stavu: na start role, vnitřní stav a konečný stav. Aktivity jsou propojeny mezi participanty komunikacemi. Komunikace mohou být rozšířeny o datové toky, a to právě o dva. Datový tok má název a směr.

4.2.8 Spojitosti mezi elementy BORM a BPMN

Na levé straně tabulky 4.3 zobrazeny na straně 84 jsou prvky z notace BORM ORD a na pravé straně jsou elementy z BPMN PSD. Bližší informace o BORM ORD lze nalézt v kapitole 3.2 Lidsky čitelný zápis modelu od strany 35 a dále u autora Knott et al. (2000), Knott et al. (2003). O BPMN lze nalézt informace od autora Chinosi & Trombetta (2012).

Konkrétně byly zohledněny poznatky z publikací (Podloucký & Pergl, 2014*a*) a (Podloucký & Pergl, 2014*b*). Publikace (Podloucký & Pergl, 2014*a*) popisuje prefixový stroj, který je použitý k formalizaci BORM ORD. Prefixový stroj je použit k

definování rozhodování u větvení. K tomuto větvení je použita v BPMN vylučovací brána XOR.

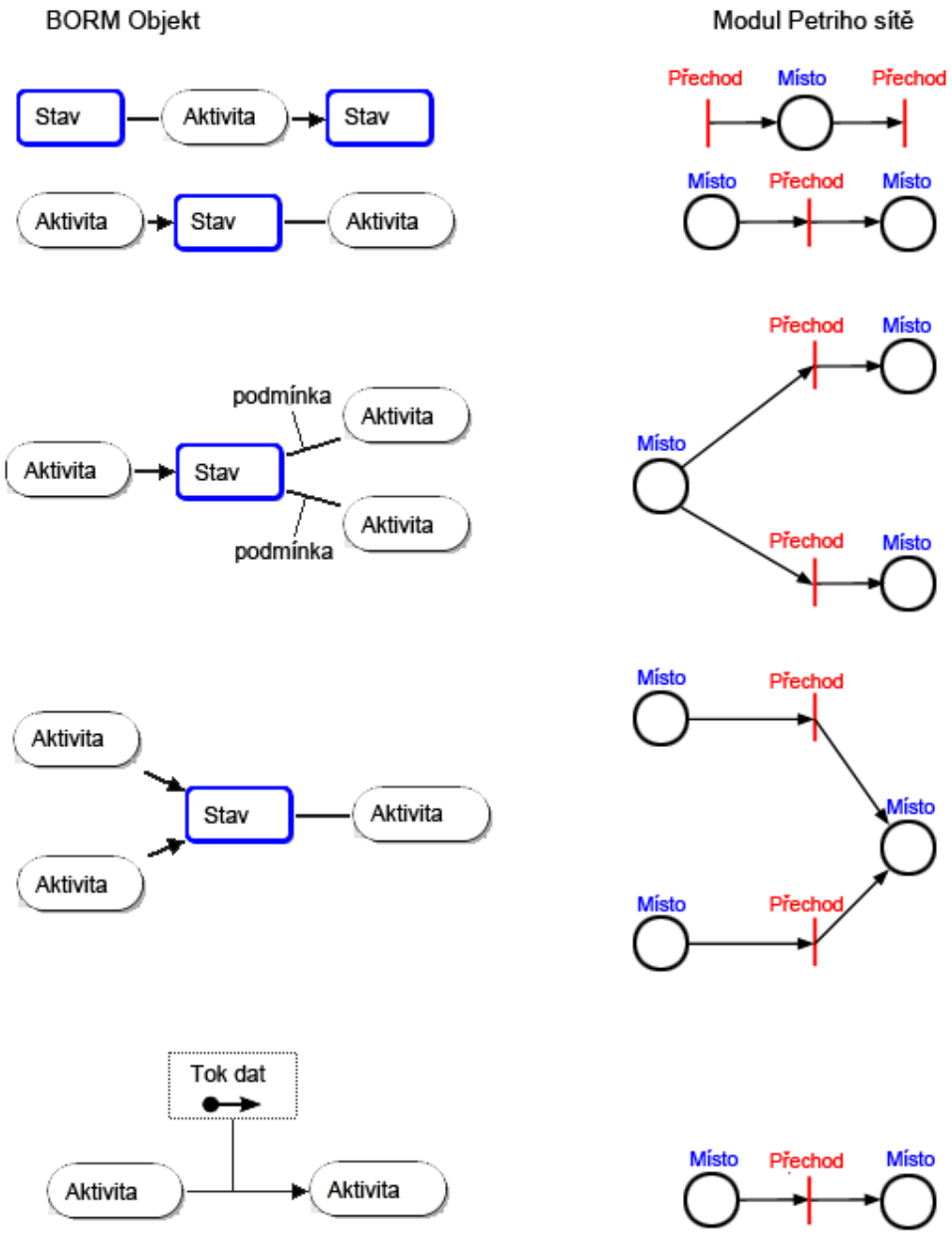
Participant	⇔	Dráha
Začátek role	⇔	Start
Konec role	⇔	Konec
Aktivita a Stav	⇔	Úloha
Přechod, Komunikace	⇔	Sekvenční tok
Datový tok	⇔	Tok zpráv

Tabulka 4.3: Tabulka zobrazení BORM ORD a BPMN PSD

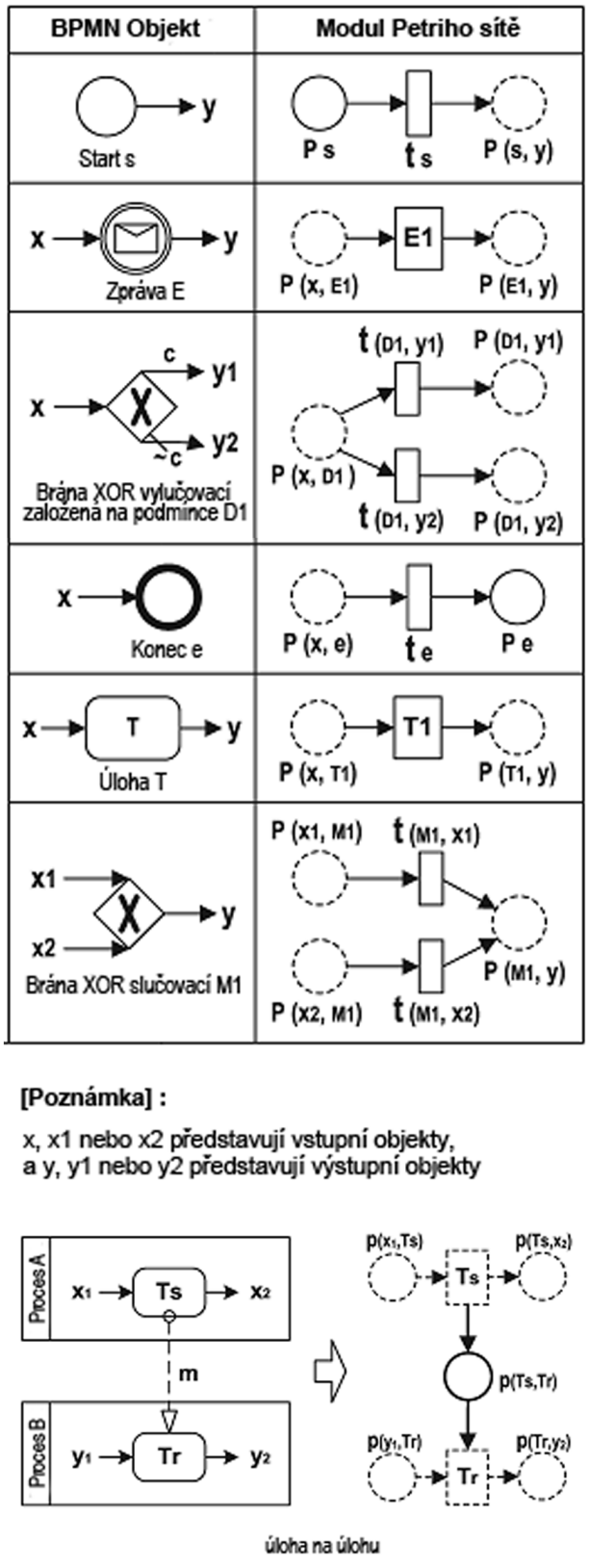
4.2.9 Formální definice spojitostí

1. Prvním krokem je nalezení spojitosti mezi konkrétními objekty modelů. Objekty modelů jsou složeny z elementů modelů.
2. Druhým krokem je formální definice spojitostí. Formální definice obsahuje syntaktickou a sémantickou spojitost. Spojitost je z modelu na Petriho síť.

Objekty BORM ORD modelu jsou ve spojitosti s Petriho sítí na obrázku 4.3 na straně 85. Objekty BPMN modelu jsou ve spojitosti s Petriho sítí na obrázku 4.4 na straně 86.



Obrázek 4.3: Spojitost BORM ORD objektů modelu s Petriho sítí podle (Papík, 2005) zpracováno autorem.



Obrázek 4.4: Spojitost BPMN objektů modelu s Petriho síťí (Dijkman & Dumas, 2007) zpracováno autorem.

4.2.10 Matematická definice spojitosti BPMN na Petriho síť

Definice 4.3. Syntaxe elementů procesu **P** modelu **M** BPMN

$$P = \langle O, F, Cond, A, E, G, T \rangle, \text{ kde} \tag{4.4}$$

Definice 4.5. Objekty **O** v modelu BPMN

$$O \in \langle A, E, G \rangle, \text{ kde} \tag{4.6}$$

A...aktivity,
E...události,
G...brány

Definice 4.7. Aktivity **A** v modelu BPMN

$$A \in \langle T \rangle, \text{ kde} \tag{4.8}$$

T...úloha

Definice 4.9. Události **E** v modelu BPMN

$$E \in \langle e^s, \quad e^i, \quad e^f \rangle, \text{ kde} \tag{4.10}$$

e^s...start (pouze jeden výskyt v diagramu),
e^i...průběžná událost,
e^f...konec (pouze jeden výskyt v diagramu)

Definice 4.11. Brány **G** v modelu BPMN

$$G \in \langle G^x, G^m \rangle, \text{ kde} \tag{4.12}$$

G^x...XOR vylučovací brána,

G^m...XOR spojovací brána

Definice 4.13. Toky v modelu **M** BPMN

$$F \subseteq \{O \times O\}, \text{ kde} \tag{4.14}$$

F...řídící spojovací toky, například sekvenční tok
O...objekty

Podmínka: $F \bigcap \{G^x \times O\} \rightarrow C, \text{kde:}$
C je funkce: $C \in \langle true, false \rangle$.

Definice 4.15. Syntaxe elementů modelu **M** BPMN, kde
M je trojice:

$$M = \langle Q, HR, F^M \rangle, \text{ kde} \tag{4.16}$$

Q...množina procesů P,
HR...spojitý graf
F^M...množina toků zpráv mezi procesy

Definice 4.17. Spojitý graf HR v modelu **M** BPMN

$$HR = \{(P_1, P_2) \in Q \times Q\}, \text{ kde} \tag{4.18}$$

Q...množina procesů P,

Definice 4.19. Množina toků zpráv F^M mezi procesy **P** v modelu **M** BPMN

$$F^M \subseteq (\bigcup_{P \epsilon Q} (T_P \bigcup e_p^f \bigcup e_p^i) \times \bigcup_{P \epsilon Q} (T_P \bigcup e_p^s \bigcup e_p^i))/U_{P \in Q}(O_P \times O_P), \text{ kde} \tag{4.20}$$

Q...množina procesů P,
T...úloha,
e^s...start (pouze jeden výskyt v diagramu),
e^i...průběžná událost,
e^f...konec (pouze jeden výskyt v diagramu)

O...objekty v modelu

Definice 4.21. Proces **P** z BPMN je kompatibilní s BORM, když:

$$F \in \langle s, f, g, x \rangle, \text{ kde:} \tag{4.22}$$

$$\{\forall s; s \in e^s, in(s) = 0 \wedge \mid out(s) \mid = 1\} \tag{4.23}$$

$$\{\forall f; f \in e^f, out(f) = 0 \wedge \mid in(f) \mid = 1\} \tag{4.24}$$

$$\{\forall g; g \in G^x, \mid in(g) \mid = 1 \wedge \mid out(s) \mid = 2\} \tag{4.25}$$

$$\{\forall g; g \in G^m, \mid out(g) \mid = 1 \wedge \mid in(s) \mid = 2\} \tag{4.26}$$

Věta 4.26.1. Pro všechny **x**, které jsou z množiny procesů **P**, existuje alespoň jedno **s**, pro které platí, že **s** je e^s. Potom existuje alespoň jedno **f**, které je e^f. Pro tyto **x** platí, že existuje tok z **s** do **x** a z **x** do **f**.

$$\forall x; x \in P, \exists s; s \in e_s, \exists f, f \in e_f, sF^*x \wedge xF^*f.^2 \tag{4.27}$$

Důkaz. Důkaz je založen na tranzitivitě relace **F**, tedy že pro všechny ⟨**s,x,f**⟩, které jsou z množiny procesů **P** platí, že pokud je tok **F** z **a** do **b** a současně také tok **F** z **b** do **c**, potom existuje tok **F** z **a** do **c**.

$$\{\forall x, \exists s, f; s, x, f \in P; sFx \text{ současně } xFf \text{ potom } sFf\} \tag{4.28}$$

■

[2] F^* je reflexivně tranzitivní uzávěr nad F, když xF^*y pokud je cesta z x do y a je reflexivní tedy xF^*x

Lemma 4.28.1. BPMN model M je kompatibilní s modelem BORM, když M je množina kompatibilních procesů P a HR je orientovaný acyklický graf.

Sémantika převodu BPMN na Petriho síť je zobrazena rovnicemi Petriho sítě, kde je místo popsáno rovnicí 4.30 na straně 90, přechod popsán rovnicí 4.32 na straně 90 a tok popsán rovnicí 4.34 na straně 91. Tyto rovnice vycházejí z poznatků Dijkman & Dumas (2007).

Kompletní komunikující procesy v BPMN jsou popsány rovnicemi 4.36 na straně 92.

Definice 4.29. Rovnice Petriho sítě popisující místo (Dijkman & Dumas, 2007) zpracováno autorem.

$$
\begin{aligned}
&P = \bigcup_{P \in Q} \langle S, E, F \rangle, \text{ kde:} \\
&S = \{p_s \quad | \quad s \in e_p^S\} \text{... start} \\
&E = \{\mathrm{p}_e \quad | \quad e \in e_p^E\} \text{... konec} \\
&F = \{\mathrm{p}_{(x,y)} \quad | \quad (x,y) \in F_p\} \text{... sekvenční tok} \\
&\text{Q...množina procesů P}
\end{aligned}
\tag{4.30}
$$

Definice 4.31. Rovnice Petriho sítě popisující přechod (Dijkman & Dumas, 2007) zpracováno autorem.

$$
\begin{aligned}
&T = \bigcup_{P \in Q} \langle T_p, G^X, G^M, S, E \rangle, \text{ kde:} \\
&T_p = \{x \quad | \quad x \in T_p \cup e_p^i \wedge in(x) \neq 0\} \\
&T_p \text{... úloha, průběžná událost} \\
&G^X = \{\mathrm{t}_{(x,y)} \quad | \quad x \in G_P^X \wedge y \in out(x)\} \\
&G^X \text{... vylučovací brána} \\
&G^M = \{\mathrm{t}_{(x,y)} \quad | \quad x \in G_P^M \wedge y \in in(x)\} \\
&G^M \text{... spojovací brána} \\
&S = \{t_s \quad | \quad s \in e_p^S\} \text{... začátek procesu} \\
&E = \{t_e \quad | \quad e \in e_p^E\} \text{... konec procesu} \\
&\text{Q...množina procesů P}
\end{aligned}
\tag{4.32}
$$

Definice 4.33. Rovnice Petriho sítě popisující tok (Dijkman & Dumas, 2007) zpracováno autorem.

$$
\begin{aligned}
F &= \bigcup_{P \in Q} \langle T_{p1}, T_{p2}, G^X_{p1}, G^X_{p2}, G^M_{p1}, G^M_{p2}, S, E \rangle, \text{ kde:} \\
T_{p1} &= \{(p_{(x,y)}, y) \quad | \quad y \in T_p \cup e^i_p \wedge in(y)\} \\
T_{p2} &= \{(y, p_{(y,z)}) \quad | \quad y \in T_p \cup e^i_p \wedge in(y) \neq 0 \wedge z \in out(y)\} \\
T_{p1,p2} &\text{... úloha, průběžná událost} \\
G^X_{p1} &= \{(p_{(x,y)}, t_{(y,z)}) \quad | \quad y \in G^X_p \wedge x \in in(y) \wedge z \in out(y)\} \\
G^X_{p2} &= \{(t_{(y,z)}, p_{(y,z)}) \quad | \quad y \in G^X_p \wedge z \in out(y)\} \\
G^X_{p1,p2} &\text{... vylučovací brána} \\
G^M_{p1} &= \{(p_{(x,y)}, p_{(y,z)}) \quad | \quad y \in G^M_p \wedge x \in in(y)\} \\
G^M_{p2} &= \{(t_{(y,x)}, p_{(y,z)}) \quad | \quad y \in G^M_p \wedge x \in in(y) \wedge z \in out(y)\} \\
G^M_{p1,p2} &\text{... spojovací brána} \\
S &= \{(p_s, t_s) \quad | \quad s \in E^S_p\} \cup \{(t_s, p_{(s,y)}) \quad | \quad s \in E^S_p \wedge y \in out(s)\} \\
&\text{S ... začátek procesu} \\
E &= \{(p_{(x,e)}, t_e) \quad | \quad e \in E^E_p \wedge e \in (e)\} \cup \{(t_e, p_e) \quad | \quad s \in E^E_p\} \\
&\text{E ... konec procesu} \\
&\text{Q ... množina procesů P}
\end{aligned}
\tag{4.34}
$$

Definice 4.35. Kompletní rovnice petriho sítě popisující komunikující procesy

(Dijkman & Dumas, 2007) zpracováno autorem.

$$
\begin{aligned}
PN =& P_M, T_M, F_M\rangle, \text{ kde:} \\
& \text{PN ... kompletní Petriho síť} \\
P_M =& P \cup P_1, \text{ kde:} \\
& \text{P ... 4.30} \\
P_1 =& \{p_{x,y} \quad | \quad (x,y) \in F^M \wedge y \notin \cup_{P \in Q} e_p^S\} \\
T_M =& T, \text{ kde:} \\
& \text{T ... 4.32} \\
F_M =& F \cup F_1 \cup F_2 \cup F_3, \text{ kde:} \\
& \text{F ... 4.34} \\
F_1 =& \{(x, p_{(x,y)}) \quad | \quad (x,y) \in F^M \wedge x \notin \cup_{P \in Q} e_p^E \wedge y \notin \cup_{P \in Q} e_p^S\} \\
F_2 =& \{(p_{(x,y)}, y) \quad | \quad (x,y) \in F^M \wedge x \notin \cup_{P \in Q} e_p^E \wedge y \notin \cup_{P \in Q} e_p^S\} \\
F_3 =& \{(t_x, p_{(x,y)}) \quad | \quad (x,y) \in F^M \wedge x \in \cup_{P \in Q} e_p^E \wedge y \notin \cup_{P \in Q} e_p^S\} \\
& \text{Q ... množina procesů P}
\end{aligned}
\tag{4.36}
$$

4.2.11 Metoda transformace Petriho sítě na konenčný automat

Podmínky z sekce 4.2.10 na straně 87 jsou tyto 4.37 na straně 92 a 4.38 na straně 92.

$$\forall g \in G^x, \mid in(g) \mid = 1 \wedge \mid out(s) \mid = 2 \tag{4.37}$$

$$\forall g \in G^m, \mid out(g) \mid = 1 \wedge \mid in(s) \mid = 2 \tag{4.38}$$

Tyto podmínky 4.37 na straně 92 a 4.38 na straně 92 jsou důležité pro transformaci Petriho sítě na stavový automat.

Stavový automat má vždy na každém přechodu jednu přicházející hranu in(t)=1 a jednu výchozí hranu out(t)=1. Toto je zapsáno rovnicí 4.39 na straně 93. Následkem

tohoto je, že nenastane souběžný běh, ale může nastat konflikt, například nedeterminismus.

$$\forall t \in T :| \, in(t) \, |=| \, out(t) \, |= 1. \tag{4.39}$$

Přechody v této transformaci dle vybraných prvků v objektech splňují tuto podmínku, jak je možné vidět na obrázku 4.3, kde je u každého přechodu pouze jedna vstupní hrana (tok) a jedna výstupní hrana.

Vybrané elementy z BPMN zaručují, že stavový automat v tomto případě transformace je konečný, neboť jsou zvoleny pro rozdělení toků vylučovací brány s podmínkou, které zaručují, že nedojde k nedeterminismu. Brány s podmínkou jsou vidět na obrázku 4.4 na straně 86. Nedeterminismus by nastal pokud by rozdělení toků nebylo podmíněné. V tomto případě jde více do detailu a rozdělení seomezuje na binární, což je vyjádřeno podmínkami 4.37 na straně 92 a 4.38 na straně 92.

4.2.12 Ekvivalence BPMN PSD a konenčného automatu

Jde o konečnou množinu prvků, ze kterých se skládá analýza. Jinými slovy v analýze jsou uvažovány případy, které jsou omezené (konečné). Jedná se tedy o konenčný počet stavů, které mohou nastat. Tudíž se jedná obecně o konečný automat. Determinismus je zajištěn podmínkou v transformaci na Petriho síť 4.39 na straně 93.

Tím, že lze BPMN popsat Petriho sítí a Petriho síť je za určitých okolností stavový automat, je možné odvodit, že BPMN PSD je konečný automat. Po použití ortogonální transformace elementů je tedy BPMN PSD model možné transformovat na konečný automat.

Formalizace BORM ORD na Mealyho konečný automat

Formalizace BORM ORD pomocí Mealyho konečného automatu je popsána v kapitole 3.2.5 na straně 54, kde je ukázáno, že objektově relační diagram (ORD) z metody BORM lze formálně popsat konečnými komunikujícími automaty.

Formalizace BPMN PSD na Moorův konečný automat

Analýzu transformace elementů BPMN na Petriho síť znázorňuje obrázek 4.4 na straně 86, kde je možné vypozorovat jednu důležitou vlastnost této transformace - tato vlastnost je závislost výstupní funkce pouze na stavu. Tato vlastnost je popsána v definici v kapitole 3.1.4 Moorova automatu na straně 21.

Myšlenka důkazu je založena na ukázání BPMN PSD jako konečného automatu. Toto je předvedeno v kapitole 4.2.10 na straně 87. Dále je pokračováno s tím, že BPMN PSD je Moorův konečný automat. Toto lze dokázat pomocí důkazu sporem. Důkaz bude ukazovat, že BPMN PSD je Mealyho automat. Toto je v rozporu s formalním zápisem BPMN PSD.

Kapitola 5

Realizační část

Tato kapitola je zaměřena na dokumentaci transformace modelu na text resp. modelu, který je reprezentován notací BORM (Business Object Relation Modelling) a textem, který je reprezentován reportem založeném na HTML.

5.1 Metoda automatizované transformace modelu na text

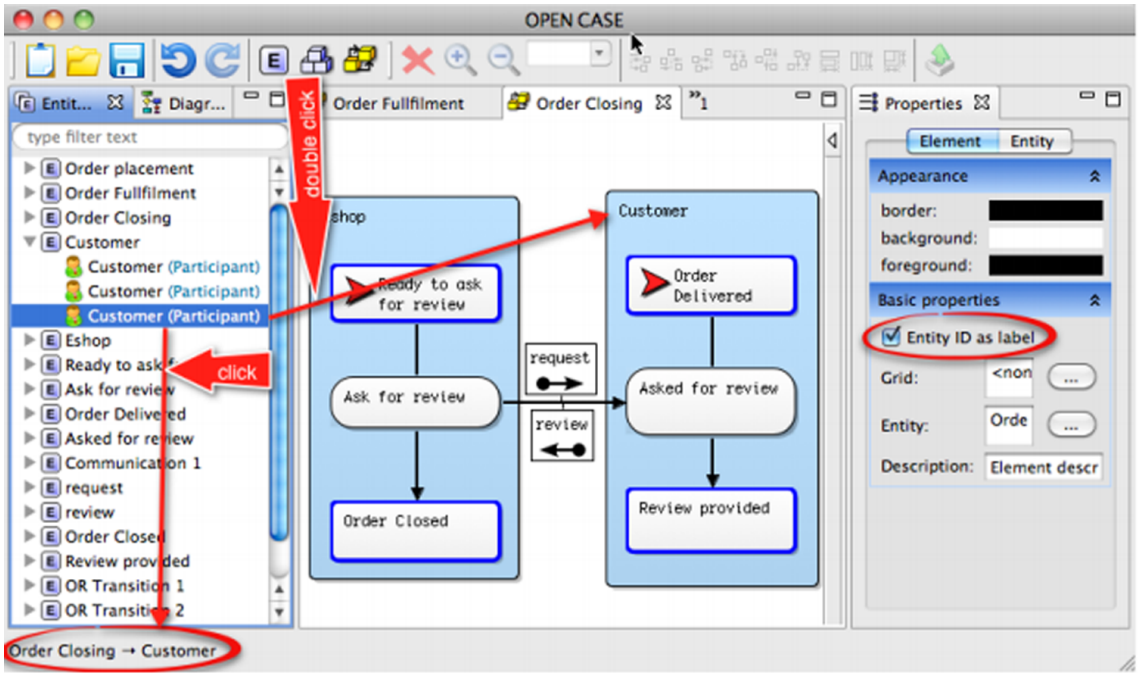
V této části je ukázáno řešení transformace modelu na text tedy generování dokumentace z BORM ORD. Obecně tato transformace podporuje inženýrství obchodních procesů. Je to kombinace příslušné metody a notace (BORM ORD) podporovaná softwarovým nástrojem (OpenCase).

Klíčovou vlastností modelovaného procesu je, že řešení není pouze diagram, ale celá znalostní báze. Tato znalostní báze může být použita pro operace, dokumentaci, rozhodování a v dalších oblastech. Zde byla použita pro automatické generování provozních manuálů. Toto je ukázáno a testováno na případových studiích v kapitole 5.3 na straně 103. Jsou ukázány dvě případové studie, které jsou na v kapitolách 5.3.3 a 5.3.2 od stránky 106.

Bylo naimplementováno rozšíření nástroje OpenCASE, který je postaven na plugin Eclipse, jak již bylo zmíněno v sekci 5.1.1 na straně 96.

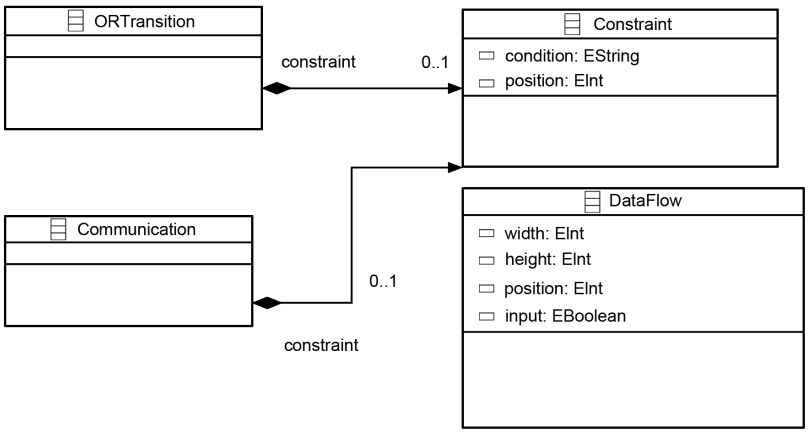
5.1.1 OpenCASE

OpenCASE je CASE nástroj navržený na podporu výzkumu v oblasti konceptuálního modelování a ontologií (opencase.net, 2011). OpenCASE byl navržen na PEF ČZU (Provozně Ekonomická Fakulta, Česká Zemědělská Univerzita v Praze) v týmu, jehož docentem byl Vojtěch Merunka, týmovým vedoucím Robert Pergl a členy byly Jakub Tůma, Marek Pícka. Tým byl zaměřen na vývoj metody BORM a jejího použití. Spolupráce byla mezi univerzitní PEF a FIT ČVUT (Fakulta Informačních Technologií, České Vysoké Učení Technické v Praze). Ukázku OpenCASE prototypu je možné vidět na obrázku 5.1 na straně 96. Nástroj je postaven na frameworku Eclipse a připraven na mnoho pokročilých možností. V nástroji byl z komplexní metody BORM naimplementován BA diagram (Business Architecture) a OR diagram (Object Relation) (Pergl & Tůma, 2012*b*).



Obrázek 5.1: Ukázka OpenCASE prototypu podle (Pergl & Tůma, 2012*b*)

OpenCASE je implementován v programovacím jazyku Java, který je podporován platformou Eclipse a různými modelovacími framework z projektu Eclipse Modeling (EMF). Eclipse Rich Client Platform (RCP) nabízí velmi silnou základnu pro systémové komponenty a platformu pro vytvoření komplexních aplikací s širokým spektrem uživatelského rozhraní (Pergl & Tůma, 2012*b*). Na obrázku 5.2 na straně 97 je znázorněna část metamodelu, který je reprezentací OR diagramu.



Obrázek 5.2: Část Ecore modelu z diagramu OR metody BORM (Pergl & Tůma, 2012*b*)

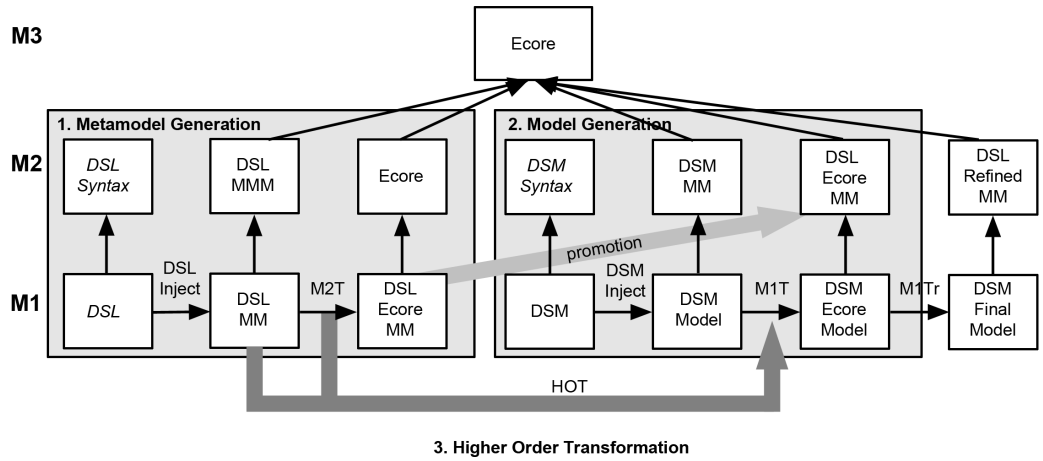
5.1.2 Realizace

Na případové studii je znázorněna transformace z modelu úrovně obchodně řídícího procesu na model provozně obchodní. Model provozní úrovně by měl být textový, aby byl srozumitelný pro každého uživatele. Je použit nástroj OpenCASE jako znalostní základna a jeho rozhraní (API) ke generování HTML stránky pro každého participanta. Tento HTML výstup je podobný (SBVR, 2008). Generování je založeno na publikaci autora Splichal et al. (2011) a tato transformace je složena z modelovacího nástroje a založena na přístupu HOT (High Order Transformation). Nástroj pro modelování OpenCASE 5.1.1 obsahuje tuto transformaci jako rozšiřitelný modul. Brambilla et al. (2009) uvádí:"*Fáze High Order Transformation adresního mapování, jak je zobrazeno na obrázku číslo 5.3 na straně 98, zajišťuje soudržnost mezi vztahy dvou technických prostorů. Tento úkol je realizován ve dvou úkolech:*

- *První transformace obdrží DSL metamodel vytvořený z modelu vygenerovaného do metamodelu (M2).*
- *Manuálně definovaný HOT derivuje M1T transformaci přeložením M2T transformace a případně analyzování struktury DSL metamodelu.*

"

Přístup HOT byl teoreticky popsán v publikaci Brambilla et al. (2009) a citován v této kapitole viz výše a ukázán na obrázku 5.3 na straně 98. Konkrétní HOT implementace je zobrazena níže na snipetu 5.1 a 5.2 na straně 98 a 98. Tato im-



Obrázek 5.3: Diagram frameworku používající HOT (Brambilla et al., 2009)

plementace přístupu HOT je použita pro generování reportu modelu OR diagramu BORM.

Listing 5.1: Hlavní funkce pro generování report z ORD diagramu

```
(defn or-report [or-diagram]
(let [name     (first (return or-diagram :entity :id))
reports (map participant-report (participants or-diagram))
notes (note-report (or-diagram))
]
(list*
(element :h1 {} (str "OR Diagram: " name))
reports
notes
)))
```

Listing 5.2: Exportní funkce pro vytvoření XHTML

```
(defn -write [this manager file]
(with-open [writer (java.io.OutputStreamWriter. (java.io.
   FileOutputStream. file) "UTF-8")]
(let [ords     (diagrams (.getProject manager) :ORDiagram)
reports (mapcat or-report ords)]
(emit (xhtml-page "XHTML Report" reports) writer :encoding
   "UTF-8"))))
```

Snipet číslo jedna 5.1 na straně 98 ukazuje hlavní funkci. Hlavní funkce se nazývá or-report a tato hlavní funkce vytváří poloviční model z vnitřní struktury reprezen-

tující OR diagram. Poloviční model je připraven pro vytvoření XHTML stránky, která je vytvořena pomocí snipetu číslo 2 5.2 na straně 98. Vnitřní struktura OR diagramu BORM je metamodel.

Na snipetu číslo 2 5.2 na straně 98, kde je zobrazena funkce, která produkuje XHTML report. Modul sloužící k vytvoření dokumentace (XHTML reportu) byl naimplementován ve funkcionálním jazyku. Funkcionální programování přináší jednoduchost rovnic (Wadler, 1992) a možnost vice informací o funkcionálním programování (Wadler, 1992).

5.2 Metoda automatizované transformace modelu na model

Tato kapitola vytváří most mezi návrháři informačních systémů a doménovými experty pomocí automatizované transformace modelů používaných v informačních systémech BORM a BPMN.

5.2.1 Transformace BORM a BPMN

V kapitole 3.2.5 Formalizace BORM ORD na straně 54 bylo ukázáno, že BORM ORD je ekvivalentní Mealyho konečnému automatu a BPMN PSD je ekvivalentní Moorovu konečnému automatu. Transformace mezi Moorovým a Mealyho automatem je známá, jak je popsáno v kapitole 3.1.4 Transformace Moorova automatu na Mealyho automat na straně 22.

Tímto byl vytvořen most pro překlenutí z elementů BPMN na Moorův automat, který lze podle známých algoritmů převést na Mealyho automat, který je popsán v kapitole 3.1.4 Mealyho automat na straně 21. A Mealyho automat je reprezentací pro BORM ORD, jak bylo uvedeno v kapitole 3.2.5 Formalizace BORM ORD na straně 54. Transformace mezi BPMN a BORM, respektive BORM a BPMN, tedy konečnými automaty Moore a Mealy respektive Mealy a Moore, je automatizovaná.

Formalizace zobrazení BPMN PSD na BORM ORD

Zobrazení BPMN PSDiagramu na BORM ORDiagram je formálně zapsáno pomocí pravidel τ_i, které jsou označeny jako transformační pravidla. Transformační pravidla τ_i jsou definována jako uspořádaná trojice, která je složena z pravidla r_i, vstupu in_i a výstupu out_i.

Definice 5.1. Formální definice pro $\tau_i = \langle r_i, in_i, out_i \rangle$, kde $r_i, in_i \in \{BPMN^{'}\}^*$ a $out_i \in \{BORM\}^*$. Obecně platí, že F^* je reflexivně tranzitivní uzávěr nad F.

Vstup in_i je z upravené množiny $BPMN^{'}$, tato množina je upravena podle autorů (Dijkman & Dumas, 2007) a (Lam, 2012) [1]. Výstup out_i je z množiny $BORM$, která je definována na obrázku 4.2 na straně 81.

$\tau = \{\tau_i\}$

Celková množina pravidel τ je složena z jednotlivých pravidel τ_i, toto lze zapsat jako $\tau_i \in \tau$. Jednotlivá pravidla transformace jsou vypsána v tabulce 5.2 na straně 100, která obsahově vyplývá z tabulky 5.1 na straně 100. Tabulka 5.1 na straně 100 je obsahově rozšířena o zavedení zkratek k jednotlivým objektům, jak je zavádí tabulka 4.3 na straně 84. Objekty zavádí obrázky 4.4 na straně 86 a 4.3 na straně 85, kde je vždy vyobrazen objekt a transformace na modul v Petriho síti.

Participant (P)	$\Leftrightarrow$	Dráha (L)
Začátek role (Z)	$\Leftrightarrow$	Start (S)
Konec role (KR)	$\Leftrightarrow$	Konec (K)
Aktivita a Stav (A)	$\Leftrightarrow$	Úloha (T)
Přechod, Komunikace (KO)	$\Leftrightarrow$	Sekvenční tok (ST)
Datový tok (DT)	$\Leftrightarrow$	Tok zpráv (E)

Tabulka 5.1: Tabulka zobrazení BORM ORD a BPMN PSD se zkratkami vytvořeno autorem.

1.)	$L \rightarrow P$	$\tau_i = \langle L \rightarrow P, L, P \rangle$
2.)	$S \rightarrow Z$	$\tau_i = \langle S \rightarrow Z, S, Z \rangle$
3.)	$K \rightarrow KR$	$\tau_i = \langle K \rightarrow KR, K, KR \rangle$
4.)	$T \rightarrow A$	$\tau_i = \langle T \rightarrow A, T, A \rangle$
5.)	$ST \rightarrow KO$	$\tau_i = \langle ST \rightarrow KO, ST, KO \rangle$
6.)	$E \rightarrow DT$	$\tau_i = \langle E \rightarrow DT, E, DT \rangle$

Tabulka 5.2: Tabulka zobrazení a transformační pravidla τ_i BPMN PSD na BORM ORD pouze zkratky vytvořeno autorem.

[1] Úprava množiny je rozebrána v kapitole diskuze 6 na straně 108

Obecná funkce definující transformaci BORM ORD na BPMN PSD

Transformaci lze popsat složením funkcí f_1 a f_2. Funkce f_1 a f_2 jsou vyjádřeny kartézkým součinem a zobrazením.

$f_1 : S \times I \to O$

$f_2 : S \times I \to S$

$f_1 \circ f_2 : S \times I \to S \times O$

Složená funkce $f_1 \circ f_2$ ukazuje, jak dochází ke skládání jednotlivých transformačních pravidel.

Algoritmus provádějící transformaci BPMN PSD na BORM ORD

Přes všechny objekty v jednotlivých drahách bazénu diagramu BPMN PSD jsou iterovány a aplikovány pravidla, která jsou definovaná v tabulce 5.2 na straně 100, tedy transformační pravidla $\tau = \{\tau_1 ... \tau_6\}$, tedy aplikujeme algoritmus zobrazený pseudokódem 5.1 na straně 101, který je složen z dalších jednotlivých algoritmů a to transformace objektu BPMN PSD na objekt BORM ORD 5.2 na straně 102, kopírování atributů jednotlivého objektu 5.3 na straně 102 a kopírování propojení na ostatní objekty 5.4 na straně 102.

Algorithm 5.1 Transformace BPMN PSD na BORM ORD

```
1:  procedure TRANSFORM
2:      Pools ← get all Pools in BPMN PSD Diagram
3:      for all Pool ∈ Pools do
4:          Lanes ← get all Lanes in Pool
5:          for all Lane ∈ Lanes do
6:              Objects ← get all Objects in Lane
7:              for all Object ∈ Objects do
8:                  convertObject
9:                  copyAttributes
10:                 copyConnections
11:             end for
12:         end for
13:     end for
14: end procedure
```

Algorithm 5.2 Transformace objektu BPMN PSD na objekt BORM ORD

```
1: procedure convertObject
2:     switch(Object.class)
3:         case(Lane) //transformační pravidlo T1
4:             ConvertedObject.class ← Participant
5:         End
6:         case(Event) //transformační pravidlo T2,3
7:             ConvertedObject.class ← Event
8:             if Event ≻ is Start OR End then
9:                 ConvertedObject.class ← State
10:            end if
11:        End
12:        case(Activity) //transformační pravidlo T4
13:            ConvertedObject.class ← Activity, State
14:            if Event ≻ is Task then
15:                ConvertedObject.class ← Activity, State
16:            end if
17:        End
18:        case(Flow) //transformační pravidlo T5,6
19:            if Flow ≻ is SequenceFlow then
20:                ConvertedObject.class ← Transition
21:            else if Flow ≻ is MessageFlow then
22:                ConvertedObject.class ← Communication
23:            end if
24:        End
25:    End
26: end procedure
```

Algorithm 5.3 Kopírování atributů jednotlivého objektu

```
1: procedure copyAttributes
2:     ConvertedObject.attributes ← Object.attributes
3: end procedure
```

Algorithm 5.4 Kopírování propojení na ostatní objekty

```
1: procedure copyConnections
2:     ConvertedObject.connections ← Object.connections
3: end procedure
```

5.2.2 Shrnutí

V této kapitole byly shrnuty teoretické předpoklady transformace a dále potom rozvinut matematický podklad pro transformační algoritmus. Matematický podklad pro algoritmus začíná formulováním zobrazení a posléze transformačními pravidly, které jsou potom uvedeny do obecné roviny. Na základě transformačních pravidel je formulován algoritmus, který provádí automatizované transformace.

Samotný algoritmus je rozdělen do hlavní části, části založené na pravidlech a pomocné části. Hlavní část transformuje objekty (entity) v modelu. Další část algoritmu popisuje transformační pravidla, která jsou definovány v tabulce transformačních pravidel. Pomocná část je složená z tzv. procedur. Pomocné procedury mají na starosti transformaci atributů a propojení mezi entitami.

5.3 Případové studie

Ověření metody probíhalo programově (kvalitativně) pomocí automatizovaných transformací nad konkrétními případovými studiemi.

Celkem bylo provedeno mezi třiceti a čtyřiceti automatizovaných transformací. Ilustrativně jsou předvedeny tyto pilotní případové studie:

1. Agendy zahraničního oddělení fakulty vysoké školy 5.3.1, která je popsána na straně 104.
2. Elektronický obchod 5.3.2, která je popsán na straně 105.
3. Metoda pachové identifikace 5.3.3, která je na straně 106.

Agendy zahraničního oddělení fakulty vysoké školy 5.3.1, která je popsána na straně 104. Výchozí model v notaci BPMN je zobrazen na obrázku 10.1 na straně 143 a jeho ekvivalentní model v notaci metody BORM je na obrázku 10.2 na straně 144.

Druhou pilotní případovou studií byl elektronický obchod 5.3.2, který je popsán na straně 105. Výchozí model v notaci BPMN je na obrázku 10.12 na straně 149 a jeho ekvivalentní model v notaci metody BORM je na obrázku 10.6 na straně 146.

Třetí případovou studií byla metoda pachové identifikace 5.3.3, která je na straně 106, jejíž ekvivalentem je model v notaci BPMN 10.18, který je zobrazen na straně 153. Po automatizované transformaci modelu v notaci BPMN a ručním dopracování je model v notaci metody BORM na obrázku 10.13, který je zobrazen na straně 150.

Metodika verifikace je uvedena v kapitole metodika prokázání správnosti algoritmu 2.3.2 na straně 9. Validace je ukázána na případových studiích, na které je odkázáno v této kapitole.

5.3.1 Případová studie agendy zahraničního oddělení fakulty vysoké školy

Tato ukázková případová studie ilustruje kategorii případových studií. Zobrazuje pouze jeden diagram z celkového počtu čtrnácti diagramů BORM ORD. Tato případová studie byla zpracována ve spolupráci[2], v rámci které bylo vytvořeno a zpracováno sedmnáct scénářů.

Případová studie agendy zahraničního oddělení fakulty vysoké školy[2] popisuje podání zprávy o cestě do zahraničí. Cestovatel po návratu ze zahraničí pomocí informačního systému zahraniční cesty zadá informace do systému, kde se formulář "Zpráva o cestě do zahraničí"vytvoří, uloží a vytiskne. Cestovatel příjme formulář, podepíše a originál předá referentovi zahraničních cest. Referent zahraničních cest příjme formulář a založí ho. Model v metodě BORM této případové studie je zobrazen na obrázku BORM model zahraniční cesty 10.2 na straně 144. Model v notaci BPMN je zobrazen na obrázku BPMN model zahraniční cesty 10.1 na straně 143. Operační manuál je vyobrazen na obrázcích Případová studie - Operační model (manuál) pro participanta IS (Informační systém) - Zahraniční cesty 10.3, Případová studie - Operační model (manuál) pro participanta Cestovatel 10.4, Případová studie - Operační model (manuál) pro participanta Referent zahraničních cest 10.5 na stranách 144, 145, 145.

[2]Případová studie byla sepsána díky spolupráci s FIT ČVUT v Praze (Fakulta informačních technologií České vysoké učení technické)

5.3.2 Případová studie elektronického obchodu

Tato ukázková případová studie ilustruje kategorii případových studií. Tato kategorie byla zpracována ve spolupráci[3], v rámci které bylo vytvořeno a zpracováno řádově desítky případových studií. Každá případová studie obsahovala okolo pěti participantů a interakci mezi těmito participanty v rámci jednoho nebo více diagramu BORM ORD.

Případová studie je založena na procesu objednávání zboží z elektronického obchodu[4]. Případová studie byla vytvořena týmovou spoluprací založenou na kurzu informační management. Kurz informační management je součástí oboru Projektové řízení na Provozně ekonomické fakultě, která je součástí České zemědělské univerzity v Praze. K demonstraci principů transformace BORM na HTML a proces spojený s objednáním zboží z elektronického obchodu je prezentován v zjednodušené verzi. Proces objednání zboží je složen spoluprací pěti participantů: Zákazník, Doručovatel, Ekonomické oddělení, Zpracování objednávky, Dodavatel. Tento proces je zobrazen v detailnější podobě na obrázku 10.6 na straně 146.

Participant zákazník objednává nalezené zboží v elektronickém obchodu, které po vyřízení objednávky zaplatí a obdrží. Tento proces je zobrazen na obrázku 10.7 na straně 147.

Participant doručovatel přijímá doklady od zboží od ekonomického oddělení a samotné zboží, provádí přepravu s cílem doručit zboží zákazníkovi, kde očekává a přijímá platbu tento proces je zobrazen na obrázku 10.8 na straně 147.

Participant ekonomické oddělení převezme od zpracování objednávky předzpracovanou objednávku, kterou strojově zpracuje a předá doklady doručovateli. Potom dokončí fakturaci, proces je zobrazen na obrázku 10.9 na straně 148.

Participant dodavatel na základě objednání připraví zboží a po přijmutí platby zboží zasílá. Posléze dokončuje objednávku, proces je zobrazen na obrázku 10.10 na straně 148.

Participant zpracování objednávky po přijetí objednávky zasílá potvrzující email

[3]Případová studie byla sepsána díky spolupráci se studenty předmětu informační management na PEF (Provozně ekonomické fakultě).

[4]Případová studie byla sepsána díky spolupráci se studenty předmětu informační management na PEF (Provozně ekonomické fakultě).

zákazníkovi a kontroluje zboží na skladě. Pokud je zboží dostupné na skladě, je vypsán požadavek na uskladněné zboží. Pokud není zboží dostupné na skladě, je objednáno u dodavatele, zaplaceno a obdrženo. Dále je zboží obdrženo a doklady jsou odeslány do ekonomického oddělení. Participant jak je zde popsán je zobrazen na obrázku 10.11 na straně 149. Dochází k dokončení celého procesu objednání zboží.

5.3.3 Případová studie metody pachové identifikace

Případová studie[5] se zaobírá procesem identifikace pachových vzorků (MPI - metoda pachové identifikace) používané v kriminalistice. Prezentován je pouze výsek procesu z této metody, protože tato metoda je velmi rozsáhla k dokumentaci. Případová studie se zaměřila na sběr pachových vzorků a jejich analýzu.

Proces MPI se skládá ze spolupráce mezi čtyřmi participant: Regionální instituce - Distribuce, Technik, Regionální instituce - Analýza, a Inspektor. Celá případová studie je zobrazena na obrázku 10.13.

Participant regionální instituce - distribuce je zaměřena na čištění, sterilizaci a poskytování přípravků pro odebírání pachových vzorků a je zobrazena na obrázku 10.14.

Participant technik si vyžádá vzorky od regionální instituce - Distribuce a provede odběr vzorků na místě činu a předá vzorky na vyhodnocení regionální instituci - Analýza a je zobrazena na obrázku 10.15.

Participant regionální instituce - Analýza převezme pachové vzorky od technika a vyhodnotí jejich schodu či neshodu a oznámí je inspektorovi, který je dále zapracovává a je zobrazena na obrázku 10.16.

Participant inspektor obdrží výsledky shody či neshody pachových vzorků a zapracovává je. Celek je zobrazen na obrázku 10.17.

[5]Případová studie byla sepsána díky spolupráci s FAPPZ (Fakulta agrobiologie potravinových a přírodních zdrojů), konkrétně s Ing. Petrem Vlasákem z Centra výcviků psů.

5.3.4 Kvantifikace případových studií

V této práci jsou uvedeny pouze ilustrativní ukázky z případových studií. Počet zpracovaných diagramů, participantů a dalších prvků při ověřování vypracovaných metod shrnuje tabulka 5.3 kvantifikace případových studií na straně 107. V tabulce jsou jednotlivé řádky kroky z OBA, která je popsána v sekci 3.2.3 na straně 47.

Tabulka 5.3: Kvantifikace případových studií

Krok OBA	Počet
Participanti	161
Procesy	78
Scénáře	49
Diagramy	43

Kapitola 6

Shrnutí výsledků a diskuze

V této kapitole jsou diskutovány výsledky předkládané disertační práce, a to konkrétně metoda automatizované transformace modelu na text a metoda automatizované transformace modelu na model.

Modely v analýze informačních systémů jsou popsány v kapitole lidsky čitelný zápis modelu 3.2 na straně 35. Tyto modely jsou součástí modelování systému a modelování sytému je součástí inženýrství požadavků. Pro inženýrství požadavků a analýzu se využívá textová forma, tedy přirozený jazyk, grafická forma nebo kombinace přirozeného jazyka a grafické formy. Pro zpracování textové formy je možné použít přístup RSLingo, který nabízí při zpracování přirozeného jazyka způsob zápisu. Tento způsob zápisu je možné strojově zpracovávat. Součástí zpracování je zkoumání úplnosti a validování zápisu, což je popsáno v kapitole RSLingo 3.1.6 na straně 33.

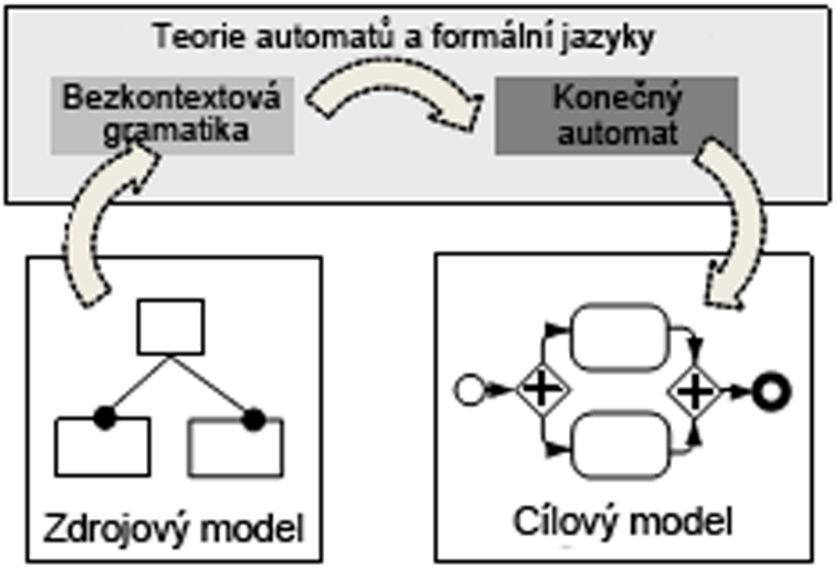
Grafická forma reprezentace analýzy informačních systémů je v podobě metody BORM, která je popsána v kapitole Business Object Relation Modeling 3.2.2 na straně 42 a notace BPMN, která je popsána v kapitole BPMN 3.2.1 na straně 35. Oba tyto zápisy grafické formy je možné vyjádřit pomocí stavových automatů. Jejich vzájemný převod je popsán v kapitole metoda automatizované transformace modelu na model 4.2 na straně 78, blíže pak v kapitole 4.2.9 na straně 84.

Oproti zmiňovaným grafickým zápisům analýzy informačních systémů je přístup, který využívá UML. Tento přístup je popsán v kapitole 3.1.5 na straně 27. Jde o přístup XIS.

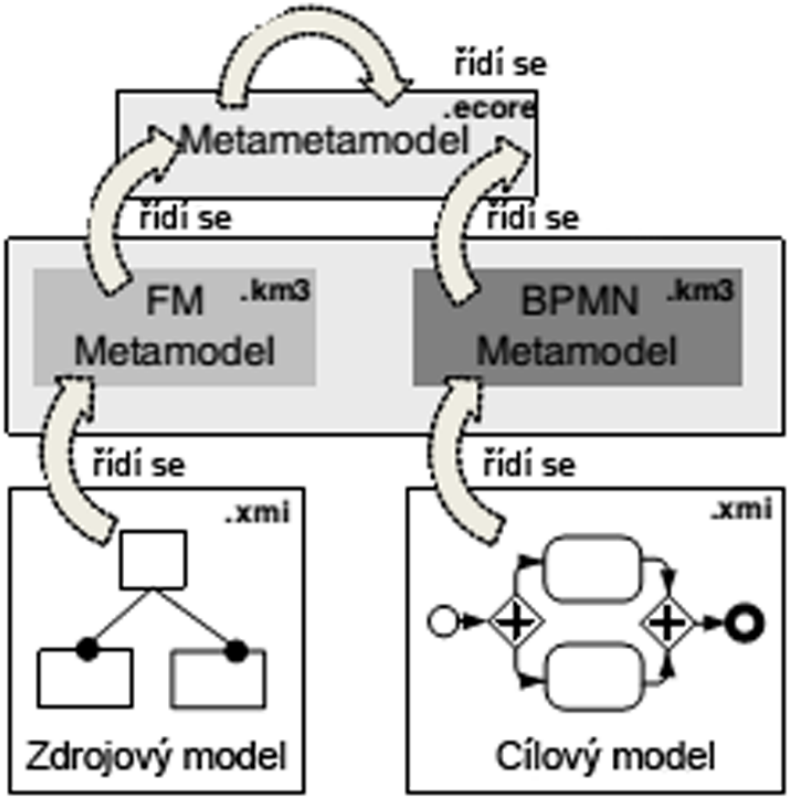
Konkrétním cílem této disertační práce, jejíž název je Metody automatizované transformace modelů v analýze informačních systému, bylo spojení standardů BPMN a metody BORM pomocí techniky do takové míry podrobnosti, aby bylo možné tyto techniky implementovat matematickým zápisem, který popisuje transformace z modelu BPMN do modelu metody BORM.

Metoda transformace modelu PSD BPMN na model ORD BORM je popsána matematickým zápisem. Tato metoda je svým přístupem ojedinělá. Nejblíže se této metodě blíží přístup autorů (Montero et al., 2008), který je zobrazen na obrázku 6.1 na straně 110. Tato metoda od autorů (Montero et al., 2008) je odvozena od přístupu Process Family Engineering (PFE) (Schnieders & Puhlmann, 2006). Přístup PFE studuje variabilitu systému bez nahlížení do detailu, při niž dojde k vytvoření modelu variabilit (FM - feature model). Z modelu FM, jak je znázorněn na obrázku 6.1 na straně 110, je patrné, jak dojde k vytvoření obchodního procesu. O přístupu PFE je blíže napsáno v publikaci (Bayer et al., 2005).

Odlišnosti mezi metodou přístupu autorů (Montero et al., 2008) a metodou přístupu využitou v předkládané doktorské disertační práci jsou popsány v kapitole 4.2. Obsah kapitoly pokračuje v matematickém popisu a důkazu přes Petriho sítě, konkrétně je formulovaná definice 4.2.11. V této definici dochází ke specifikaci obecného přístupu automatického mapování. Doktorská disertační práce rozšiřuje obecný přístup mapování od autorů (Montero et al., 2008) o matematický zápis transformace. Dále je přes Petriho sítě dokázáno, že tento specifický přístup přes Mealyho automat je možný. Realizace samotné automatizované transformace je vyřešena pomocí algoritmu, který je specifikovaný 5.2.1. Tímto je potvrzena první hypotéza, která je formulována na straně 2.



a. Systematický přístup k mapování



b. Prototyp mapování založený na MDD
(Model data driven - Data řízená modelem)
transformace modelu na model

Obrázek 6.1: Obdobný přístup transformace (Montero et al., 2008) zpracováno autorem

Podcílem této disertační práce byla analýza současného stavu v oblasti metody BORM, BPMN a transformací modelů, která je obsahem literární rešerše této práce. Ta je vypracována v kapitole 3 na straně 10. Obsahem literární rešerše je jak lidsky čitelný zápis modelu v kapitole 3.2 na straně 35, tak i strojově čitelný zápis modelu v kapitole 3.3 na straně 63 a přístupy k transformacím v kapitole 3.4 na straně 65, které lze provádět s modely.

V této disertační práci byly vytvořeny metody, pomocí kterých se bude transformovat analytický model podle jedné metody na model podle druhé metody. Toto je pospáno v kapitole 4.2 na straně 78. Této kapitole předchází vývojový krok, který je popsán v kapitole v kapitole 4.1 na straně 75.

Otázkou je, zda nedůslednost a neúplnost můžou být zmenšeny pomocí automatizace při vyjádření požadavků ve vhodném jazyce a odpovídajícím nástroji (da Silva, 2014).

Odůvodnění proč byla zvolena metoda BORM je, metoda BORM má podle autorů publikace z roku 2003 (Knott et al., 2003) následující výhody, jak je uvedeno v tabulce 3.1 na straně 45.

Základem je předpoklad, že musí dojít k modelování obchodních procesů. BORM poskytuje konzistentní přístup a notaci pro analýzu a návrh podnikového prostředí, které je zdrojem požadavků na systém. BORM navazuje na procesně orientovaný přístup, který kombinuje s čistě objektovým paradigmatem. Procesně orientovaný přístup vede ke komplexnější a rychlejší analýze při řešení problému. Koncoví uživatelé nebo obecněji zúčastněné strany problémové domény jsou schopni velmi rychle pochopit diagrmay používané v BORM.

6.1 Metoda automatizované transformace modelu na text

Ekvivalence Participanta z notace BORM ORD s tabulkou v dokumentaci reportu je patrná z tabulky 4.2. Obě tyto entity modelu lze vyjádřit jako prvky konečného automatu.

V této části bylo ukázáno řešení transformace modelu na text tedy generování

dokumentace z BORM ORD. Obecně tato transformace podporuje inženýrství obchodních procesů. Je to kombinace příslušné metody a notace (BORM ORD) podporovaná softwarovým nástrojem (OpenCase).

Klíčovou vlastností modelovaného procesu je, že řešením není pouze diagram, ale celá znalostní báze. Tato znalostní báze může být použita pro operace, dokumentaci, rozhodování a v další oblasti. V této doktorské disertační práci byla použita pro automatické generování provozních manuálů.

V předkládané doktorské disertační práci bylo implementováváno rozšíření nástroje OpenCASE, který je postaven na plugin Eclipse, jak již bylo zmíněno v sekci 5.1.1 na straně 96.

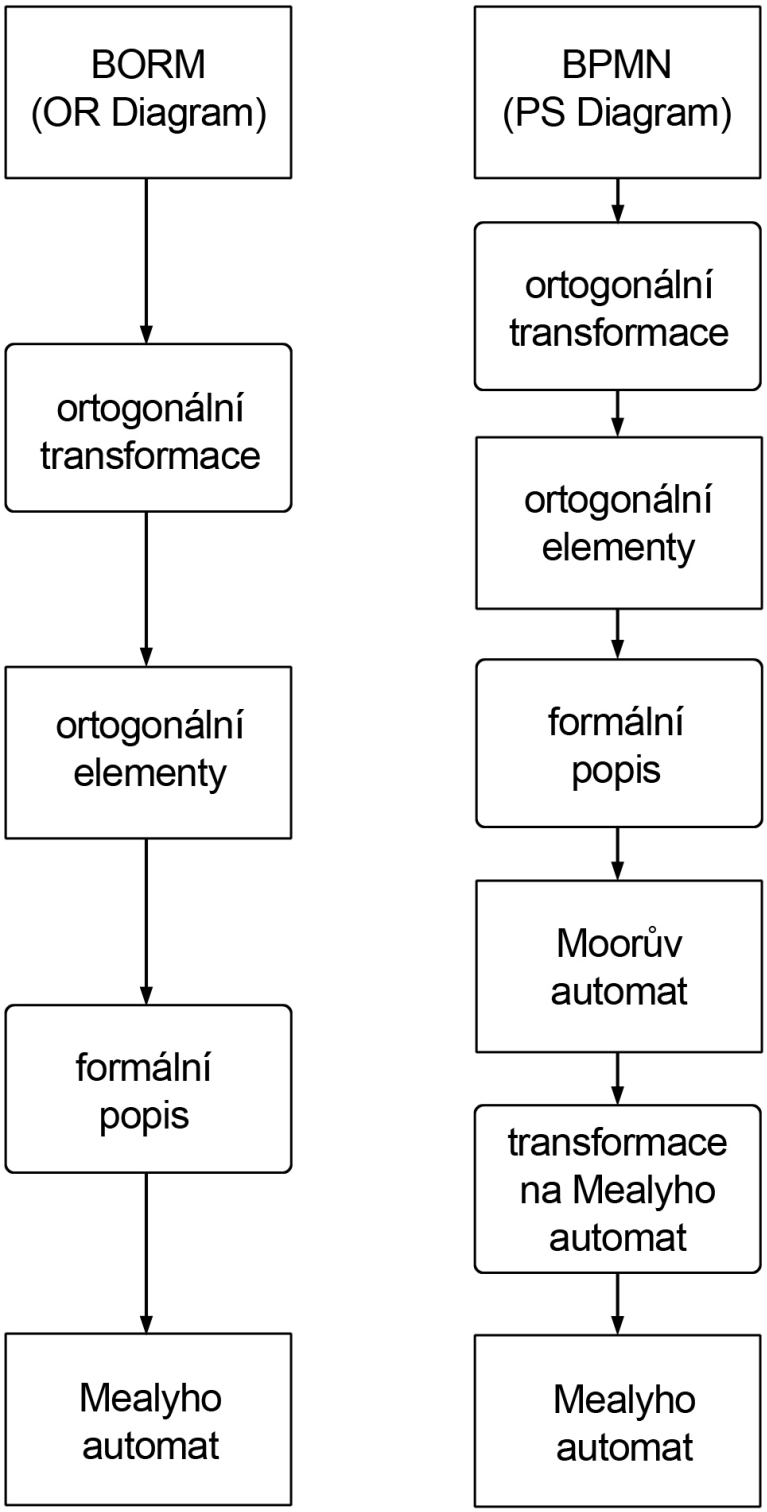
6.2 Metoda automatizované transformace modelu na model

Tato doktorské disertační práce je ojedinělá v zaměření transformace modelů, kdy velice populární a známý model v notaci BPMN převádí na méně známý a starší model metody BORM.

Rozlišné přístupy v transformacích mezi technikami z jedné na druhou jsou ukázány například v (Figueira & Aveiro, 2014) a (Dijkman & Dumas, 2007). Publikace (Figueira & Aveiro, 2014) ukazuje transformaci z techniky DEMO na BPMN. Transformace z DEMO na BPMN používá tzv. action rule syntax, což je syntaxe akčních pravidel. Publikace (Dijkman & Dumas, 2007) precizně popisuje BPMN model na Petriho sítě. Tato publikace sloužila této práci jako jeden z inspirujících materiálů. Představená metoda transformace není doposud známá. Tento fakt dokládá publikace (Montero et al., 2008), která popisuje způsob transformace založený také na konečných automatech, ale ve spojení s kontextovou gramatikou, jak je ukázáno na obrázku 6.1.

Ze strany metody BORM je transformace dosažena v publikaci (Tuma & Hanzlık, 2015), která navazuje na publikaci (Moravec, 2014) vyplývající z (Moravec & Papík, 2010). Ze strany notace BPMN je transformace ukázána postupným odvozením přes metamodel a Petriho sítě v kapitole 4.2 na straně 78. Celý tento postup je matematicky popsán, jak ze strany metody BORM v kapitole 4.1 na straně 75, tak

ze strany notace BPMN v kapitole 4.2 na straně 78. Grafické znázornění postupu je zobrazeno na obrázku 6.2 na straně 113.



Obrázek 6.2: Grafické znázornění metody transformace BPMN na BORM, autor

Tedy metoda automatizované transformace modelů BPMN na BORM využívající transformaci přes Petriho sítě a konečné automaty je ojedinělá a nová. Na základě nalezených spojitostí v oblasti Petriho sítí a konečných automatů byl vytvořen a ověřen algoritmus 5.1 na straně 101, který umožňuje provádět automatizované transformace z modelu BPMN na model metody BORM.

Elementy popsané v kapitole Elementy BPMN a BORM 4.2.4 na straně 79 jsou diskutovány níže.

Elementy pro BPMN PSD jsou:

- Proces
- Aktivity
- Tok
- Událost
- Úloha
- Sekvenční tok
- Tok zpráv
- Konec
- Průběžná
- Start
- Brána
- Bazén

Je nutné dodat, že elementy BPMN byly vybrány s ohledem na publikované výstupy autorů (Dijkman & Dumas, 2007) a (Lam, 2012), které se věnují ve své práci redundancí elementů v notaci BPMN. Notace BPMN je uvedena například v publikaci (Chinosi & Trombetta, 2012). Úplná definice notace BPMN je uvedena v specifikaci ve verzi 1.0 (OMG, 2006) a v verzi 2.0 (OMG, 2012).

Elementy pro BORM ORD jsou:

- Participant
- Událost
- Stav
- Aktivita
- Přechod mezi stavy
- Podmínka
- Datový tok
- Komunikace
- Typ stavu
    - Start
    - Vnitřní
    - Konec

Při samotném výběru elementů použitých v souvislosti s metodou BORM byly konkrétně zohledněny poznatky z publikací (Podloucký & Pergl, 2014*a*) a (Podloucký & Pergl, 2014*b*). Publikace (Podloucký & Pergl, 2014*a*) popisuje prefixový stroj, který je použitý k formalizaci BORM ORD. Prefixový stroj je použit k definování rozhodování u větvení. K tomuto větvení je použita v BPMN vylučovací brána XOR.

V této práci byla dokázána automatizovaná transformace modelů podle popsané metodiky oproti autorům Montero et al. (2008), kteří využívají k transformaci modelů bezkontextovou gramatiku a teorii automatů. Přístup autorů Montero et al. (2008) je zaměřen na transformaci modelu funkcí na model procesní. Oproti tomu, přístup této doktorské disertační práce je zaměřen na překlenutí rozdílů mezi modely, které jsou procesní, tedy událostmi založený model BPMN a stavově založený model BORM.

V modelu BORM ORD je hlavní prvek, z které je složen proces participanta, stav (Knott et al., 2006). Oproti tomu v modelu BPMN PSD je hlavní prvek v procesu událost (Lam, 2010).

Kapitola 7

Závěr a doporučení

V této disertační práci jsou shrnuty vědecké poznatky v oblasti věnující se tématu Metody Automatizovaných Transformací Modelů v Analýze Informačních Systémů.

Poznatky v oblasti modelů jsou popsány v kapitole lidsky čitelný zápis modelu a kapitole strojově čitelný zápis modelu.

Kapitola lidsky čitelný zápis modelu 3.2 na straně 35 a dalších podkapitolách věnující se notaci BPMN 3.2.1 na straně 35 a notaci, metodu BORM 3.2.2 na straně 42.

Kapitola strojově čitelný zápis modelu 3.3 na straně 63 uvádí stručný úvod do Domain model 3.3.1 na straně 63 a EMF 3.3.2 na straně 63.

Poznatky v oblasti transformací jsou popsány v kapitole Transformace modelu na model kapitola 3.4 na straně 65, kde jsou rozebrány možnosti přístupů k transformacím. V této kapitole jsou uvedeny návaznosti na MDA 3.4.1 k nalezení na straně 65.

Důležitým přehledem současného stavu jsou kategorie přístupů transformací 3.4.2 na straně 66. Tento přehled je dále propracován v následujících kapitolách od strany 67 do strany 5 kromě.

7.1 Závěr

Výsledkem této doktorské disertační práce je algoritmus, který provádí automatizované transformace z modelu PSD BPMN na model ORD metody BORM.

První hypotéza je formulována: Je možné propojit notaci BPMN a metodu BORM přes Mealyho automat. Základem první hypotézy byla myšlenka, že společným bodem je Mealyho automat.

Druhá hypotéza byla formulována: Je možné dosáhnout Mealyho automatu ze strany notace BPMN a metody BORM.

Závěrem lze říci, že první hypotéza a z ní vycházející druhá hypotéza jsou dosažitelné, pokud je použit postup graficky zobrazený na obrázku 6.2 na straně 113. Tedy podrobněji Mealyho automat není přímo dosažitelný z obou směrů, tedy jak ze strany metody BORM a ze strany BPMN, ale je nutné provést transformaci ve větvi strany BPMN, který je ekvivalentní automatu Moore. Tedy transformace je z automatu Moore na automat Mealy, který je předpokládaným výchozím bodem pro propojení obou stran.

7.2 Přínos disertační práce

V této disertační práci, jejíž tématem je Metody Automatizované Transformace Modelů v Analýze Informačních Systému, je hlavním přínosem vytvoření metody, pomocí které se algoritmizovaným způsobem transformuje analytický model na následné modely podle zásad MDA. Obecným cílem disertační práce bylo přispět k holistickému přístupu vývoje informačním systémům.

Konkrétním cílem této disertační práce bylo spojení standardů BPMN a metody BORM pomocí techniky do takové míry podrobnosti, aby bylo možné tyto techniky implementovat matematickým zápisem, který bude popisovat transformace z modelu BPMN do modelu metody BORM.

Podcílem této disertační práce byla analýza současného stavu v oblasti metody BORM, BPMN a transformací modelů, která je obsahem literární rešerše této práce.

Formulovaná metoda se zabývá metodou BORM a notací BPMN a jejich vzájemnou vazbou.

Samotný převod na výchozí bod - tedy Mealyho automat je ze strany metody BORM přímý. Oproti tomu převod na Mealyho automat ze strany notace BPMN není přímý, ale je nutné jej dosáhnut přes transformaci automatu Moore na Mealy. Tento krok je nutný pro dosažení společného výchozího bodu - Mealyho automatu. Protože notace BPMN je v samotné podstatě Moorův automat.

Postup propojení metody BORM a notace BPMN přes Mealyho automat je následující. Na straně notace BPMN dochází k výběru z specifikace procesního diagramu (PSDiagram - Process Structure Diagram) pomocí ortogonální transformace elementů tohoto diagramu na ortogonální elementy. Tyto transformované elementy jsou mapovány na prvky metody BORM ORDiagramu. Transformované elementy jsou formálně popsány jak syntakticky tak sémanticky pomocí petriho sítě, odkud je odvozeno formálně na konečný automat. V tomto případě se jedná o Moorův automat.

Na straně metody BORM dochází k výběru z specifikace objektově relačního diagramu (ORDiagram - Object Relation Diagram) pomocí ortogonální transformace elementů tohoto diagramu na ortogonální elementy. Tyto transformované elementy jsou formálně mapovány na prvky konečného automatu. V tomto případě jde přímo o Mealyho automat.

Formalismus propojení BPMN a metody BORM je dokončen navrženým algoritmem, který je popsán v kapitole transformace BORM a BPMN.

V kapitole metodický přístup je popsána verifikace a validace navrženého formalismu (algoritmu).

Validace formalismu byla provedena na případových studiích. Ověření metody probíhalo programově (kvalitativně) pomocí automatizovaných transformací nad konkrétními případovými studiemi. Celkem bylo provedeno okolo třiceti automatizovaných transformací z nichž jsou uvedeny tři v této práci pouze ilustrativně.

Tato transformace vytváří most mezi návrháři informačních systémů a doménovými experty pomocí automatizované transformace modelů používaných v informačních systémech BORM a BPMN.

Výstup této doktorské disertační práce rozšiřuje stávající metodu BORM o automatizovanou transformaci modelů. Metoda BORM je založená na postupných transformacích v jednotlivých fázích analýzy, návrhu, a vývoje informačního systému. Při využití automatizované transformace oproti manuální dojde k redukci chyb, které jsou vytvořené při manuální transformaci. Došlo tedy k vylepšení již propracované metody BORM. Byl vytvořen most mezi BPMN a BORM.

7.3 Doporučení

Doporučení pro další pokračování výzkumu je rozšíření transformace na další modely z notace BPMN a metody BORM. Například jde rozšířit o model Business architecture (BA) diagram z metody BORM.

Kapitola 8

Seznam zkratek

Zkratky použité v textu disertační práce jsou vysvětleny v tabulce 8.1 na straně 121.

Tabulka 8.1: Vysvětlení zkratek

Zkratka	Význam	Vysvětlení
API	Application Programming Interface	Rozhraní pro programování aplikací
ARIS	Architecture of Integrated Information Systems	Metodika pana profesora Scheera pro tzv. reengineering
AS-IS	Stávající stav	Definuje stávající stav podnikového procesu
ATL	Atlas Transformation Language	Transformační jazyk atlas
AVS	Architektura výpočetních systémů	Předmět vyučovaný na PEF ČZU
B2B	Business to business	Je označení pro obchodní vztahy mezi obchodními společnostmi
BORM	Business Object Relation Modeling	Metoda pro pokrytí všech fází vývoje softwaru
BPM	Business Process Management	Procesní řízení
BPMN	Business Process Model And Notation	Grafické znázorňování podnikových procesů pomocí procesních diagramů. Standard pro modelování podnikových procesů
BPR	Business Process Reingeneering	Reengineering podnikových procesů
CASE	Computer-aided software engineering	Počítačem podporované softwarové inženýrství
CMMI	Capability Maturity Model Integration	Model kvality organizace práce určený pro vývojové týmy
CRC	Class-Responsibility-Collaborator	Třída-zodpovědnost-spolupráce jsou nástrojem používaném při návrhu objektově orientovaného softwaru
CSA	Computer Systems Architecture	Předmět vyučovaný na PEF ČZU
DFD	Data Flow Diagrams	Diagramy datového toku
DKIS	Database and Knowledge Information Systems	Předmět vyučovaný na PEF ČZU
DKM	Datové a znalostní modelování	Předmět vyučovaný na PEF ČZU
DZIS	Databázové a Znalostní Informační Systémy	Předmět vyučovaný na PEF ČZU
DZM	Data and Knowledge Modelling	Předmět vyučovaný na PEF ČZU
EJB	Enterprise Java Beans	Jsou řízené, serverové komponenty umožňující modulární tvorbu podnikových aplikací
EMF	Eclipse Modeling Framework	Je framework pro vytváření nástrojů a jiných aplikací založených na strukturovaném datovém modelu

Zkratka	Význam	Vysvětlení
IDEF3	Integrated DEFinition for Process Description Capture Method	Metoda sloužící k zachycení popisu procesu, jak pracuje část systému
JAD	Joint Application Design	Je proces používaný k prototypování životního cyklu sbírání obchodních požadavků pro vývoj softwaru
JMI	Java Metadata Interface	Rozhraní definující metadata pro jazyk Java
KTSW	Komponentová tvorba SW	Předmět vyučovaný na PEF ČZU
MDA	Model-Driven Architecture	Modelem řízená architektura
MOF	Meta Object Facility	Je množina standardů rozhraní, které mohou být použity k definování a manipulaci množiny metamodelů a jejich příslušných
OBA	Object Behavioral Analysis	Je studium a modelování domény v kontextu stanovených cílů
OCL	Object Constraint Language	Je jazyk pro popis pravidel, které se aplikují na UML
OMG	Object management Group	Skupina zabývající se standardy
ORD	Object-Relationship-Diagram	Diagram z BORM
PIM	Platform Independent Model	Model nezávislý na platformě
PSMs	Platform Specific Models	Model závislý na platformě
SI	Software Engineering	Softwarové inženýrství
STD	State-Transition Diagram	Stavový diagram s přechody
TO-BE	Budoucí stav	Užívá se ve spojení s AS-IS
TQM	Total Quality Management	Komplexní řízení kvality
TRL	Transformation Rule Language	Transformační-generativní gramatika
UML	Unified Modeling Language	Je v softwarovém inženýrství grafický jazyk například pro navrhování a dokumentaci programových systémů
USI	Universita della Svizzera italiana	Zahraniční univerzita
XMI	XML Metadata Interchange	Standard pro výměnu metadat pomocí XML
XML	eXtensible Markup Language	Je obecný značkovací jazyk
XPath	XML Path Language	Je jazyk, pomocí kterého lze adresovat části XML dokumentu

Kapitola 9

Reference

Agrawal, A., Karsai, G. & Lédeczi, Á. (2003), An end-to-end domain-driven software development framework, *in* 'Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications', ACM, pp. 8–15.

Aguilar-Saven, R. S. (2004), 'Business process modelling: Review and framework', *International Journal of production economics* **90**(2), 129–149.

Akehurst, D. & Kent, S. (2002), A relational approach to defining transformations in a metamodel, *in* '≪ UML≫ 2002—The Unified Modeling Language', Springer, pp. 243–258.

Akehurst, D., Kent, S. & Patrascoiu, O. (2003), 'A relational approach to defining and implementing transformations between metamodels', *Software and Systems Modeling* **2**(4), 215–239.

Alcatel, Softeam, Thales, TNI-Valiosys & et al., C. C. (2003), 'Mof query/-views/transformations, revised submission. omg document: ad/03-08-05'. OMG Document: ad/03-08-05.

Andries, M., Engels, G., Habel, A., Hoffmann, B., Kreowski, H.-J., Kuske, S., Plump, D., Schürr, A. & Taentzer, G. (1999), 'Graph transformation for specification and programming', *Science of Computer programming* **34**(1), 1–54.

AndroMDA (2003), 'Andromda 2.0.3, july 2003, http://www.andromda.org'. http://www.andromda.org.

Architecture, T. M.-D. (2003), 'The model-driven architecture, guide version 1.0.1, omg document: omg/2003-06-01'. OMG Document: omg/2003-06-01.

ArcStyler (2004), 'Arcstyler 4.0, september 2004, http://www.arcstyler.com/'. http://www.arcstyler.com/.

Babcsanyi, I. (2000), 'Equivalence of mealy and moore automata', *Acta Cybern.* **14**(4), 541–552.

Banks, J. et al. (1998), *Handbook of simulation*, Wiley Online Library.

Bassett, P. (1997), 'Framing software reuse: Lessons from the real world'. Prentice Hall, Inc., 1997.

Bayer, J., Buhl, W., Giese, C., Lehner, T., Ocampo, A., Puhlmann, F., Richter, E., Schnieders, A., Weiland, J. & Weske, M. (2005), Process family engineering: Modeling variant-rich processes, Technical report, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-PlattnerInstitute.

Bézivin, J., Dupé, G., Jouault, F., Pitette, G. & Rougui, J. E. (2003), First experiments with the atl model transformation language: Transforming xslt into xquery, *in* '2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture', number 1, p. 50.

Bider, I. (2005), 'Choosing approach to business process modeling-practical perspective', *Journal of Conceptual Modeling* **34**, 1–16.

Bird, S., Klein, E. & Loper, E. (2009), *Natural language processing with Python*, "O'Reilly Media, Inc.".

b+m ArchitectureWare (2006), 'b+m architectureware, generator framework, http://www.architectureware.de'. http://www.architectureware.de.

Bondy, J. A. & Murty, U. (2008), 'Graph theory, volume 244 of graduate texts in mathematics'.

Booch, G., Rumbaugh, J. & Jacobson, I. (1999), 'The unified modeling language user guide', *Reading, UK: Addison Wesley* .

Brambilla, M., Fraternali, P. & Tisi, M. (2009), A transformation framework to bridge domain specific languages to mda, *in* 'Models in Software Engineering', Springer, pp. 167–180.

Brand, D. & Zafiropulo, P. (1983), 'On communicating finite-state machines', *Journal of the ACM (JACM)* **30**(2), 323–342.

Braun, P. & Marschall, F. (2003), 'The bi-directional object-oriented transformation language.'. Technical Report, Technische Universität München, TUM-I0307, May 2003.

Cabot, J., Pau, R. & Raventós, R. (2010), 'From uml/ocl to sbvr specifications: A challenging transformation', *Information systems* **35**(4), 417–440.

CBOP, D. & IBM (2003), 'Cbop, dstc, and ibm. mof query/views/transformations'. Revised Submission. OMG Document: ad/03-08-03.

CC, M. (2007), 'Compuware corporation and sun microsystems, mof query/-views/transformations, revised submission. omg document: ad/03-08-07'. OMG Document: ad/03-08-07.

Chandy, K. M. (2006), 'Event-driven applications: Costs, benefits and design approaches', *Gartner Application Integration and Web Services Summit* **2006**.

Chinosi, M. & Trombetta, A. (2012), 'Bpmn: An introduction to the standard', *Computer Standards & Interfaces* **34**(1), 124–134.

Cleaveland, C. (2001*a*), 'Program generators with xml and java.'. http://www.craigc.com/pg/.

Cleaveland, J. C. (2001*b*), *Program Generators with XML and JAVA*, Prentice-Hall.

Committee, I. C. S. S. E. S. & Board, I.-S. S. (1998), Ieee recommended practice for software requirements specifications, Institute of Electrical and Electronics Engineers.

ČSN, E. (2006), '9000', *Systémy managementu jakosti–Základy, zásady a slovník* .

Czarnecki, K. & Helsen, S. (2003), Classification of model transformation approaches, *in* 'Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture', Vol. 45, USA, pp. 1–17.

Czarnecki, K. & Helsen, S. (2006), 'Feature-based survey of model transformation approaches', *IBM Systems Journal* **45**(3), 621–645.

da Silva, A. R. (2003), The xis approach and principles, *in* 'Euromicro Conference, 2003. Proceedings. 29th', IEEE, pp. 33–40.

da Silva, A. R. (2014), Specqua: Towards a framework for requirements specifications with increased quality, *in* 'International Conference on Enterprise Information Systems', Springer, pp. 265–281.

Davis, A. (2013), *Just enough requirements management: where software development meets marketing*, Addison-Wesley.

de Almeida Ferreira, D. & Rodrigues da Silva, A. (2013), Rsl-pl: A linguistic pattern language for documenting software requirements, *in* 'Requirements Patterns (RePa), 2013 IEEE Third International Workshop on', IEEE, pp. 17–24.

de Almeida Ferreira, D. & Silva, A. R. d. (2012), Rslingo: An information extraction approach toward formal requirements specifications, *in* 'Model-Driven Requirements Engineering Workshop (MoDRE), 2012 IEEE', IEEE, pp. 39–48.

Delta (2004), 'Frame processor angie, delta software technology'. http://www.d-s-t-g.com/neu/pages/pageseng/et/common/techn_angie_frmset.htm.

Dietz, J. L. (2006), *What is Enterprise Ontology?*, Springer.

Dijkman, R. M. & Dumas, Marlon a Ouyang, C. (2007), 'Formal semantics and analysis of bpmn process models using petri nets', *Queensland University of Technology, Tech. Rep* .

El Emam, K. & Koru, A. G. (2008), 'A replicated survey of it software project failures', *Software, IEEE* **25**(5), 84–90.

Emrich, M. (2003), 'Generative programming using frame technology.'. http://www.geocities.com/mslerm/xframerf.htm.

Figueira, C. & Aveiro, D. (2014), A new action rule syntax for demo models based automatic workflow process generation (demobaker), *in* 'Advances in Enterprise Engineering VIII', Springer, pp. 46–60.

FPL (2004), 'Frame-processing-language, http://sourceforge.net/projects/fpl'. http://sourceforge.net/projects/fpl.

Frankel, D. S. (2003), *Model Driven Architecture Applying Mda*, John Wiley & Sons.

FUUT-je (2005), 'Fuut-je, hosted at the eclipse generative model transformer (gmt) project website, http://dev.eclipse.org/viewcvs/indextech.cgi/ checkout /gmthome/download/index.html'. http://dev.eclipse.org/viewcvs/indextech.cgi/ checkout /gmthome/download/index.html.

FXVCL (2005), 'Xml-based variant configuration language, http://fxvcl.sourceforge.net/'. http://fxvcl.sourceforge.net/.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1994), *Design patterns: elements of reusable object-oriented software*, Pearson Education.

Ganesan, E. (2010), 'Process modeling – art and science of university of understanding business. infosys techonologies ltd.'. http:// papers.ssrn.com /sol3 /papers.cfm?abstract_id=1929713.

Gerber, A., Lawley, M., Raymond, K., Steel, J. & Wood, A. (2002), Transformation: The missing link of mda, *in* 'Graph Transformation', Springer, pp. 90–105.

Giaglis, G. M. (2001), 'A taxonomy of business process modeling and information systems modeling techniques', *International Journal of Flexible Manufacturing Systems* **13**(2), 209–228.

Gill, A. (1962), 'Introduction to the theory of finite-state machines'.

Girault, C. & Valk, R. (2003), *Petri nets for systems engineering: a guide to modeling, verification, and applications*, Springer.

Goldberg, A. & Rubin, K. S. (1995), 'Succeeding with objects. decision frameworks for project management', *Reading, Mass.: Addison-Wesley,— c1995* **1**.

Gouda, M. G., Manning, E. G. & Yu, Y.-T. (1984), 'On the progress of communication between two finite state machines', *Information and control* **63**(3), 200–216.

Group, G. P. (2004), 'Generative programming group, http://www.generative-programming.org'.

Habáň, J. & Sodomka, P. (2004), 'Jak rozumí systémové integraci dodavatelé informačních systémů?', *SYSTEMS INTEGRATION* p. 229.

Helm, R., Johnson, R., Vlissides, J. & Gamma, E. (2002), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

Hliněný, P. (2007), *Teorie Grafů*, verze 0.99 edn, FI MU.
URL: *http://www.fi.muni.cz/ hlineny/Vyuka/GT/Grafy-text07.pdf*

Holzmann, G. J. (1990), Design and validation of protocols, *in* 'Tutorial Computer Networks and ISDN Systems', Citeseer.

Hommes, B. (2004), 'The evaluation of business process modeling techniques.'. Dissertation thesis, Delft University of Technology, 2004.

Hopcroft, J. E., Motwani, R. & Ullman, J. D. (2001), 'Introduction to automata theory, languages, and computation', *ACM SIGACT News* **32**(1), 60–65.

Indulska, M., Green, P., Recker, J. & Rosemann, M. (2009), Business process modeling: Perceived benefits, *in* 'Conceptual Modeling-ER 2009', Springer, pp. 458–471.

Jamda (2003), 'Jamda: The java model driven architecture 0.2, may 2003, http://sourceforge.net/projects/jamda/'. http://sourceforge.net/projects/jamda/.

Jensen, K. (1987), Coloured petri nets, *in* 'Petri nets: central models and their properties', Springer, pp. 248–299.

Jeston, J. & Nelis, J. (2014), *Business process management*, Routledge.

JET (2002), 'Java emitter templates (jet). part ofthe eclipse modeling framework, see jet tutorial by remko pompa at http://eclipse.org/articles/articlejet2/jet_tutorial2.html'. http://eclipse.org/articles/ArticleJET2/jet_tutorial2.html.

Juric, M. B., Mathew, B. & Sarang, P. G. (2006), *Business Process Execution Language for Web Services: An Architect and Developer's Guide to Orchestrating Web Services Using BPEL4WS*, Packt Publishing Ltd.

Kadlec, V. (2004), *Agilní programování: metodiky efektivního vývoje softwaru*, Computer press.

Kettinger, W. J., Teng, J. T. & Guha, S. (1997), 'Business process change: a study of methodologies, techniques, and tools.', *MIS quarterly* **21**(1).

Knott, R., Merunka, V. & Polak, J. (2003), 'The borm methodology: a third-generation fully object-oriented methodology', *Knowledge-Based Systems* **16**(2), 77–89.

Knott, R., Merunka, V. & Polak, J. (2006), 'The borm method: A third generation object-oriented methodology', *Management of the Object-Oriented Development Process* pp. 337–360.

Knott, R. P., Merunka, V. & Polak, J. (2000), Process modeling for object oriented analysis using borm object behavioral analysis, *in* 'Requirements Engineering, IEEE International Conference on', IEEE Computer Society, pp. 7–7.

Ko, R. K., Lee, S. S. & Lee, E. W. (2009), 'Business process management (bpm) standards: a survey', *Business Process Management Journal* **15**(5), 744–791.

Koch, N. (2007), 'Classification of model transformation techniques used in uml-based web engineering', *IET software* **1**(3), 98–111.

Korherr, B. & List, B. (2007), Extending the epc and the bpmn with business process goals and performance measures., *in* 'ICEIS (3)', pp. 287–294.

Korthaus, A. (1998), Using uml for business object based systems modeling, *in* 'The Unified Modeling Language', Springer, pp. 220–237.

Kovitz, B. L. (1999), *Practical software requirements: a manual of content and style*, Manning Greenwich.

Krasner, G. E. & Pope, S. T. (1988), 'A cookbook for using the model-view controller user interface paradigm in smalltalk-80', *J. Object Oriented Program.* **1**(3), 26–49.
URL: *http://dl.acm.org/citation.cfm?id=50757.50759*

Krogstie, J. (2012), Perspectives to process modeling–a historical overview, *in* 'Enterprise, Business-Process and Information Systems Modeling', Springer, pp. 315–330.

Kruchten, P. (2004), *The rational unified process: an introduction*, Addison-Wesley Professional.

Lam, V. (2012), 'Foundation for equivalences of bpmn models', *Theoretical and Applied Informatics* **24**(1), 33.

Lam, V. S. (2010), 'Formal analysis of bpmn models: a nusmv-based approach', *International Journal of Software Engineering and Knowledge Engineering* **20**(07), 987–1023.

Lindsay, A., Downs, D. & Lunn, K. (2003), 'Business processes—attempts to find a definition', *Information and software technology* **45**(15), 1015–1019.

Marschall, F. & Braun, P. (2003), Model transformations for the mda with botl, *in* 'Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications', pp. 25–36.

Mbarki, S. & Erramdani, M. (2009), 'Model-driven transformations: from analysis to mvc 2 web model.', *International Review on Computers & Software* **4**(5).

Mealy, G. H. (1955), 'A method for synthesizing sequential circuits', *Bell System Technical Journal* **34**(5), 1045–1079.

Merunka, V. (2008), *Objektové modelování*, Alfa Nakladatelství.

Merunka, V. (2012*a*), Fsm-based object-oriented organization modeling and simulation, *in* 'Advanced Information Systems Engineering Workshops', Springer, pp. 398–412.

Merunka, V. (2012*b*), 'Seminář metodologie vědy 2012, pef Czu praha'. PEF CZU Praha.

Merunka, V., Pergl, R. & Tůma, J. (2015), Borm-ii and uml as accessibility process in knowledge and business modelling, *in* 'New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering', Springer, pp. 1–6.

Merunka, V. & Tůma, J. (2013), 'Normalization rules of the object-oriented data model', *Lecture Notes in Electrical Engineering* **152 LNEE**, 1077–1089. cited By (since 1996)0.
URL: *http://www.scopus.com /inward /record.url?eid=2-s2.0-84866628713&partnerID=40&md5=67760949e81f97c7482d1e80986069ae*

Molhanec, M. & Merunka, V. (2011), 'Borm: Agile modelling for business intelligence', *Business Intelligence and Agile Methodologies for Knowledge-Based Organizations: Cross-Disciplinary Applications* pp. 120–130.

Montero, I., Pena, J. & Ruiz-Cortes, A. (2008), From feature models to business processes, *in* 'Services Computing, 2008. SCC'08. IEEE International Conference on', Vol. 2, IEEE, pp. 605–608.

Moore, E. F. (1956), 'Gedanken-experiments on sequential machines', *Automata studies* **34**, 129–153.

Moravec, J. (2014), *Orchestrace a choreografie procesů v BORM*, 2014, PEF ČZU.

Moravec, J. & Papík, M. (2010), Transformace borm-petriho sít' s využitím supervize, *in* 'Sborník konference Objekty', pp. 1–15.

Murata, T. (1989), 'Petri nets: Properties, analysis and applications', *Proceedings of the IEEE* **77**(4), 541–580.

OMG (1999), '"white paper on the profile mechanism", version 1.0, omg document ad/99-04-07. omg uml working group.'.

OMG (2003*a*), 'Interactive objects and project technology, mof query/views/transformations, revised submission. omg document: ad/03-08-11, ad/03-08-12, ad/03-08-13'. OMG Document: ad/03-08-11, ad/03-08-12, ad/03-08-13.

OMG (2006), 'Bpmn 1.0: Omg final adopted specification, february 6, http://www.bpmn.org/documents/omg final adopted bpmn 1-0 spec 2006-02-01.pdf'. http://www.bpmn.org/Documents/OMG Final Adopted BPMN 1-0 Spec 2006-02-01.pdf.

OMG (2011), 'Omg® (2011) formal specifications, `http://www.omg.org/spec/`'. http://www.omg.org/spec/.

OMG (2012), 'Omg bpmn. bpmn 2.0. bpmn specification. [online] 2011. [citace: 2012-07-01]. dostupné z: http://www.omg.org/spec/bpmn/2.0/'. http://www.omg.org/spec/BPMN/2.0/.

OMG, M. (2003*b*), 'Omg, meta object facility 1.4, omg document: formal/02-04-03'. OMG Document: formal/02-04-03.

OMG, U. (2001), 'Object management group, the unified modeling language 1.5, omg document: formal/03-03-01'. OMG Document: formal/03-03-01.

OMG, X. (2002), 'Omg. xml metadata interchange (xmi) version1.2 http://www.omg.org/cgi-bin/doc?formal/2002-01-01'.

opencase.net (2011), 'Opencase tool'. http://opencase.net/.

OptimalJ (2006), 'Optimalj 3.0, user's guide, http://www.compuware.com/products/optimalj'. http://www.compuware.com/products

Ould, M. A. (2005), *Business Process Management: a rigorous approach*, BCS, The Chartered Institute.

Ouvans, C., Dumas, M., Ter Hofstede, A. H. & van der Aalst, W. M. (2006), From bpmn process models to bpel web services, *in* 'Web Services, 2006. ICWS'06. International Conference on', IEEE, pp. 285–292.

P. Braun, F. M. (2003), The bi-directional object-oriented transformation language, *in* 'Technical Report, Technische Universität München', Technische Universität München, pp. TUM–I0307.

Papík, M. (2005), Diploma thesis - nástroje procesního modelování (tools for process modeling), Master's thesis, Czech University of Life Sciences in Prague.

Peng, W. & Puroshothaman, S. (1991), 'Data flow analysis of communicating finite state machines', *ACM Transactions on Programming Languages and Systems (TOPLAS)* **13**(3), 399–442.

Pergl, R. & Tůma, J. (2012*a*), Opencase–a tool for ontology-centred conceptual modelling, *in* 'Advanced Information Systems Engineering Workshops', Springer, pp. 511–518.

Pergl, R. & Tůma, J. (2012*b*), Opencase– a tool for ontology-centred conceptual modelling, *in* M. Bajec & J. Eder, eds, 'Advanced Information Systems Engineering Workshops', Vol. 112 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, pp. 511–518.
URL: *http:dx.doi.org10.1007978-3-642-31069-0_42*

Phalp, K. T. (1998), 'The cap framework for business process modelling', *Information and Software Technology* **40**(13), 731–744.

Podloucký, M. & Pergl, R. (2014*a*), The prefix machine–a formal foundation for the borm or diagrams validation and simulation, *in* 'Enterprise and Organizational Modeling and Simulation', Springer Berlin Heidelberg, pp. 113–131.

Podloucký, M. & Pergl, R. (2014*b*), Towards formal foundations for borm ord validation and simulation, Vol. 2, pp. 315–322. cited By 0.

Pohl, K. (2010), *Requirements engineering: fundamentals, principles, and techniques*, Springer Publishing Company, Incorporated.

Polák, J., Merunka, V. & Carda, A. (2003), *Umění systémového návrhu: Objektově orientovaná tvorba informačních systémů pomocí původní metody BORM*, Grada Publishing.

Puntambekar, A. (2008), *Formal Languages And Automata Theory*, Technical Publications.

QVT-Partners (2003), 'Qvt-partners. mof query/views/transformations'. Revised Submission, OMG Document: ad/2003-08-08.

Recker, J., Rosemann, M., Indulska, M. & Green, P. (2009), 'Business process modeling-a comparative analysis.', *Journal of the Association for Information Systems* **10**(4).

Robertson, S. & Robertson, J. (2012), *Mastering the requirements process: Getting requirements right*, Addison-wesley.

SBVR, O. (2008), 'Semantics of business vocabulary and business rules (sbvr), version 1.0'.

Schnieders, A. & Puhlmann, F. (2006), 'Variability mechanisms in e-business process families.', *BIS* **85**, 583–601.

Shallit, J. O. & Shallit, J. (2009), *A second course in formal languages and automata theory*, Vol. 179, Cambridge University Press Cambridge.

Shlaer, S. & Mellor, S. J. (1992), *Object lifecycles: modeling the world in states*, Yourdon Press.

Smith, H. & Fingar, P. (2003), *Business process management: the third wave*, Vol. 1, Meghan-Kiffer Press Tampa.

Sommerville, I. (2010), *Software Engineering*, 9th edn, Pearson.

Sommerville, I. & Sawyer, P. (1997), *Requirements engineering: a good practice guide*, John Wiley & Sons, Inc.

Splichal, P., Pergl, R. & Pícka, M. (2011), Borm model transformation, *in* 'P. Šplíchal (2011) Model Transformation, Agrarian perspectives proceeding of the 20 th International Scientific Conference, Prague', pp. 423–430.

Steinberg, D., Budinsky, F., Merks, E. & Paternostro, M. (2008), *EMF: eclipse modeling framework*, Pearson Education.

Struska, Z. & Merunka, V. (2007), Borm points-new concept proposal of complexity estimation method., *in* 'ICEIS (3)', pp. 580–586.

Struska, Z. & Pergl, R. (2009), Borm-points: Introduction and results of practical testing, *in* 'Enterprise Information Systems', Springer, pp. 590–599.

The Software Patterns Series (1996 - 2002), Addison Wesley.

Tůma, J. (2011), Diploma thesis - generování webových aplikací na základě doménového modelu (model driven generation of web applications), Master's thesis, Czech Technical University in Prague.

Tůma, J. (2013*a*), 'Borm-ii a bpmn v provozně ekonomických procesech'.

Tůma, J. (2013*b*), 'Methods of automated model transformations in information system analysis', *Procedia Technology* **8**, 612–617.

Tůma, J., Merunka, V. & Pergl, R. (2014), Object-oriented fsm-based approach to process modelling, *in* 'Modern Trends and Techniques in Computer Science', Springer, pp. 597–606.

Tůma, J., Picka, M. & Hanzlik, P. (2015), 'Generated report of the ord borm model', *Acta Informatica Pragensia* **2015**(1), 30–43.

Tuma, J. & Hanzlık, P. (2015), 'Automated model transformation method from borm to bpmn', *Applied Mathematical Sciences* **9**(116), 5769–5777.

UIML.org (2000), 'Uiml.org. uiml v2.0 draft specification'.

Varró, D., Varró, G. & Pataricza, A. (2002), 'Designing the automatic transformation of visual languages', *Science of Computer Programming* **44**(2), 205–227.

Velocity (2003), 'Velocity 1.3.1, the apache jakarta project, march 2003, http://jakarta.apache.org/velocity/'. http://jakarta.apache.org/velocity/.

Řepa, V. (2007), *Podnikové procesy-procesní řízení a modelování-2., aktualizované a rozšířené vydání*, Grada Publishing as.

W3C, X. (1999), 'W3c, xml path language version 1.0, november 1999, http://www.w3.org/tr/xpath'. http://www.w3.org/TR/xpath.

Wadler, P. (1992), The essence of functional programming, *in* 'Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages', ACM, pp. 1–14.

West, B. D. (2002), *Introduction to Graph Theory*, Pearson Education.

Willink, E. D. (2003), Umlx: A graphical transformation language for mda, *in* 'Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications', University of Twente, pp. 13–24.

Wright, D. R. (2005), 'Finite state machines', *CSC215 Class Notes Prof. David R. Wright website, N. Carolina State University* .

www.merriam webster.com (2014), 'http://www.merriam-webster.com/dictionary/methodology'. http://www.merriam-webster.com/dictionary/methodology, 2014.

XDE, R. (2005), 'Rational xde, http://www.rational.com/products/xde'. 2016: https://www.ibm.com/software/rational.

XDoclet (2006), 'Xdoclet - attribute oriented programming, http://xdoclet.sourceforge.net/'. http://xdoclet.sourceforge.net/.

Kapitola 10

Přílohy

Rejstřík

10.1 O autorovi

Autor této disertační práce byl hlavní řešitel dvouletého grantu (2012-2013) na PEF ČZU a navrhovatel grantů k řešení v rámci celouniverzitní interní grantové agentury a fakultní interní grantové agentury.

Autor této rešerše působil jako lektor na Provozně ekonomické fakultě České Zemědělské Univerzity po dobu pěti semestrů v kurzech: v zimním semestru 2011/2012 Komponentová tvorba softwaru (SW), v letním semestru 2011/2012 Datové a znalostní modelování, v zimním semestru 2012/2013 Komponentová tvorba softwaru (SW), v letním semestru 2012/2013 zahraniční studium na Universita della Svizzera italiana, kde úspěšně absolvoval semestr studia v anglickém jazyce, v zimním semestru 2013/2014 kurzy: Architektura výpočetních systémů, Komponentová tvorba softwaru (SW) a Computer Systems Architecture, v letním semestru 2013/2014 kurzy: Databázové a znalostní Informační systémy (IS), Data and Knowledge Modelling, Database and Knowledge Information Systems (IS). V zimním semestru 2014/2015 vedl cvičení předmětů ICT Theory vyučovaném v anglickém jazyce a cvičení předmětu Informační management vyučovaném v českém jazyce. V letním semestru 2014/2015 byl na pracovní stáži v Lisabonu na Inesc ID.

Autor se zúčastnil zahraničních konferencí v průběhu prvního a druhého ročníku studia.

V druhé polovině prvního roku studia se autor zúčastnil mezinárodní konference CAISE 2012, na které byla participace na workshopu-podsekci EOMAS, která se konala ve městě Gdaňsk (Polsko). Sborník z této konference vyšel u nakladatelství Springer.

V závěru druhého ročníku se autor zúčastnil mezinárodní konference HAICTA 2013 (Korfu), která byla za účasti participantů z celé Evropy zaměřena na aplikaci informatiky v zemědělství. Sborník z této konference vyšel u nakladatelství Elsevier.

Dále se autor zúčastnil elektronických konferencí CISSE, které jsou pravidelně pořádány Univerzitou Bridgeport (New York). Tato konference se světovou/mezinárodní účastí probíhala pomocí videokonferenčních hovorů. Sborník z této konference vyšel u nakladatelství Springer.

Publikované konference

- CISSE 2011 - 7th International Joint Conferences on Computer, Information, Systems Sciences, & Engineering
- CAISE 2012 - 24th International Conference on Advanced Information Systems Engineering
- HAICTA 2013 - 6th International Conference on Information and Communication Technologies in Agriculture, Food and Environment
- CISSE 2013 - 9th International Joint Conferences on Computer, Information, Systems Sciences, & Engineering
- CSOC 2014 - 3rd Computer Science On-line Conference
- ERIE 2014 - 11th conference Efficiency and Responsibility in Education

Publikované výstupy

- (Merunka & Tůma, 2013)
- (Pergl & Tůma, 2012*a*)
- (Tůma, 2013*b*)
- Účast na (Tůma, 2013*a*)
- (Tůma et al., 2014)
- (Tůma et al., 2015)
- (Tuma & Hanzlık, 2015)

Výsledky vědecké práce autora o průběhu vypracování disertace byly průběžné uveřejňovány ve vědeckých publikacích.

2011 Tato publikace (Merunka & Tůma, 2013), která byla uveřejněna na konferenci v roce 2011, se věnuje normalizací objektově orientovaných dat. Existuje pouze několik přístupů jak normalizovat objektově orientovaná data. Přístup u objektově orientované databáze se nazývá normalizace tříd. V publikaci jsme představili

přístup normalizace objektově orientovaného konceptuálního modelu založeném na diagramu tříd UML. První část publikace popisuje současný stav v oblasti formálních metod použitých pro objektově orientované datové modelování. Druhá část publikace popisuje čtyři normální pravidla, které jsou založeny na zkušenostech autorů a modifikovaném přístupu Amber-Beck . Tyto normální pravidla jsou představeny na konkrétních příkladech. Naše metoda byla použita při výuce na několika univerzitách. Taktéž se používala pro návrh databází při vývoji softwarových projektů. V současnosti byl započat vývoj CASE nástroje.

2012 OpenCASE je originální CASE nástroj, který podporuje konceptuální modelování a byl prezentován v této publikaci (Pergl & Tůma, 2012*a*). CASE nástroj byl vyvinut během výzkumu zaměřeného na konceptuální modelování ontologií. Toto poskytuje silný důraz na výrazy a jejich vztahy, zatímco podporuje standartní notace (v současnosti je podporována metoda BORM, další notace jsou v plánu). Nástroj je postaven na otevřené architektuře pluginů založeném na platformě Eclipse, která vytváří nástroj modulární a rozšířitelný. Znalostní báze modelů je přístupná přes API a tedy je možné doimplementovat verifikace, rozlišné kalkulace (statistiky), transformovat modely na výstupy (reporty) a provádět vnitřní transformace (například normalizace). Samotná architektura nástroje je také v publikaci krátce zmíněna.

2013 V této publikaci (Tůma, 2013*b*) autor představuje samotný PhD projekt, jehož cílem je vytvořit metodu, která bude provádět automatické transformace z jednoho modelu na model druhý v analýze informačních systémů. Obecně jde o překlenutí mezery mezi business modely a informačními analytickými modely. Tato metoda bude implementována a prokázána. Metoda transformace bude mít dopad na vyvíjený nástroj OpenCASE. Metody budou založeny na současných teoriích, ale budou zpracované v doposud neznámé formě. Přístup metody bude založen na modelem řízené architektuře (MDA). V této publikaci (Tůma, 2013*b*) jsou popsány přístupy transformací model na model a model na text.

2014 V publikaci (Tůma et al., 2014) je představen přístup založený na kombinaci konečných automatů a objektově orientovaném přístupu. Hlavní myšlenkou této publikace je modelování obchodních požadavků a vývoj softwaru. Publikace je

rozdělena na tři části. Motivace a diskuze je o potřebě propojení dvou obchodních požadavků a softwarového inženýrství, myšlenka procesního modelování a modelování obchodních situací jako konečných automatů je druhou částí. Třetí částí je zobrazení navrženého přístupu na modely BPMN a UML. Zobrazení poskytuje zajímavé nové poznatky vycházející z navrhovaného přístupu. Tento přístup je založen na naší zkušenosti s našimi projekty z praxe.

2015 V této publikaci (Merunka et al., 2015) jsou představeny dvě znalostní modelovací techniky, které mohou být použitý jako nástroje ke koordinaci komunikace mezi výzkumnými pracovníky a uživateli. Publikace je zaměřena na použití obecného přístupu UML a inovativního přístupu BORM jako komunikační standard mezi výzkumnými projekty. První část této publikace popisuje framework, který vystihuje hlavní vlastnosti obou notací, metamodel a teoretické poznatky a jejich výhody a nevýhody. Je uveden praktický příklad ze zemědělství, venkovského a organizačního prostření modelové domény. Tyto inovativní procesy v obou přístupech jsou aplikovány na stejné podnikové procesy a je vyhodnocen dopad na výzkumníky a uživatele. Hlavní část je zaměřena na transformaci model na model založenou na BORM. Transformace je v souladu se standardy OMG: UML a SBVR (Semantics of Business Vocabulary and Rules). Můj předchůdce pracoval na transformaci modelu BORM na model UML. Tato práce následuje práci Petra Šplíchal a jde dále. Tato transformace bude zasazena do modelovacího nástroje a bude založena na přístupu HOT (High Order Transformation). Cílem tohoto výzkumu je dosáhnout výstupu dokumentace jako je SBVR a překlenou mezeru mezi zúčastněnými stranami obchodními a stranou vývoje.

Další je výstup (Tůma et al., 2015) zaměřený na dokumentaci transformace model na model, kde první model je reprezentovaný BORM (Business Object Relation Modelling) a druhý model je reprezentován reportem založeným na HTML. Nástroj OpenCASE je použit v tomto výstupu. Tento nástroj je rozšířitelný a pomocí rozšíření je řešena transformace a generování reportů, kterým porozumí i obchodní zastoupená strana. Report je založený na HTML, který je přenositelný a rozšířitelný. Dále je představena případová studie, která ukazuje transformaci z procesního řídícího obchodního modelu na model operační úrovně. Model operační úrovně by měl být textový a pro každého účastníka.

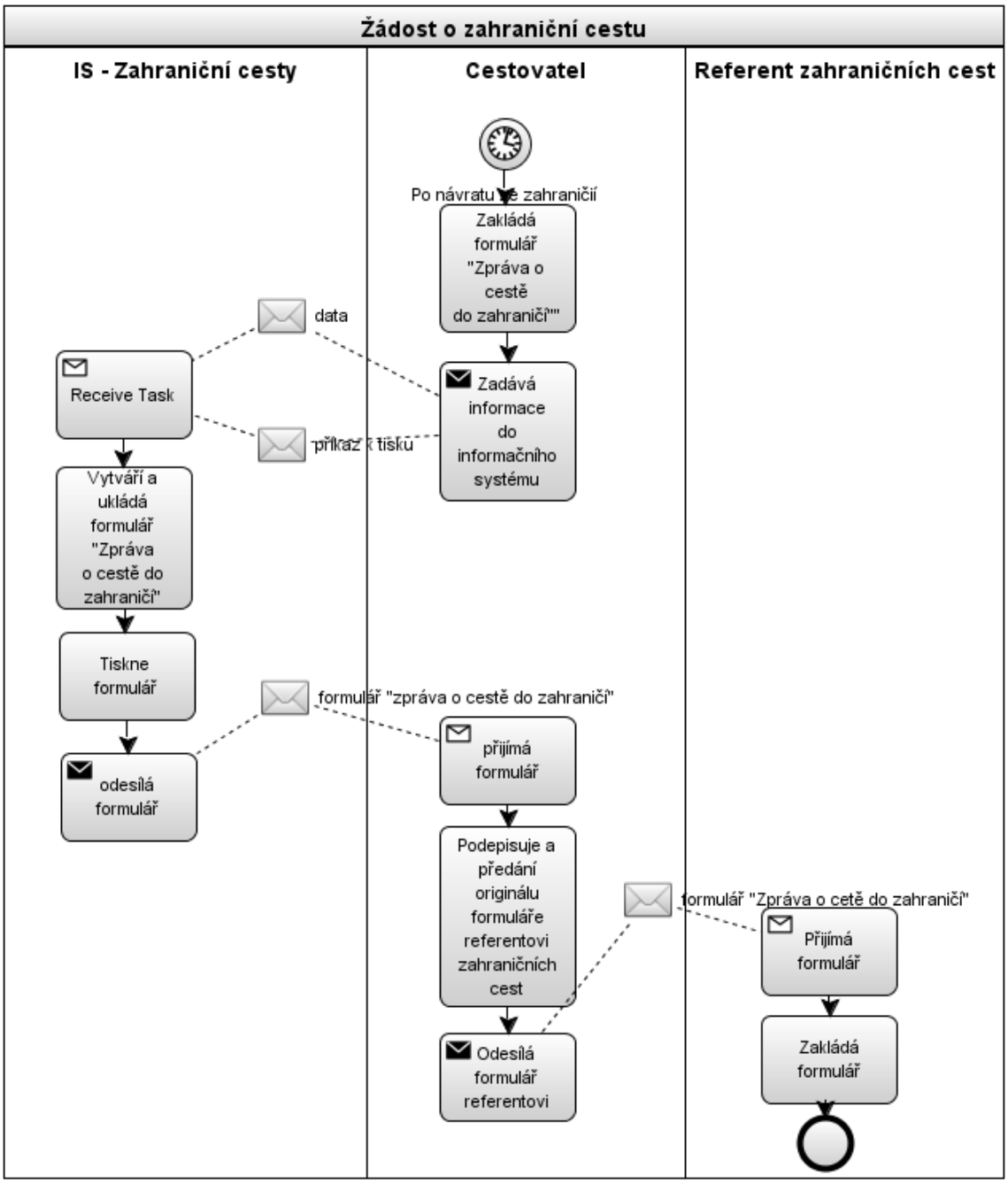
Třetí výstup (Tuma & Hanzlık, 2015) představuje přístup založený na kombinaci

konečných automatů a objektově orientovaného přístupu. Myšlenka modelování proces jako konečných automatů a představení mapování s BPMN (Business Process Modelling Notation). V této publikaci se zaměřujeme na použití BPMN a BORM (Business Object Relation Modelling). Jdeme více do hloubky a propojujeme BORM a BPMN. V hlavní část zmiňujeme transformaci modelu na model založeném na BORM. Tato transformace jde hlouběji a navazuje na předchozí prezentovanou transformaci model na text, kde vstupem je BORM a výstupem dokumentace obdobná SBVR. Návrh transformace je mezi BORM a BPMN.

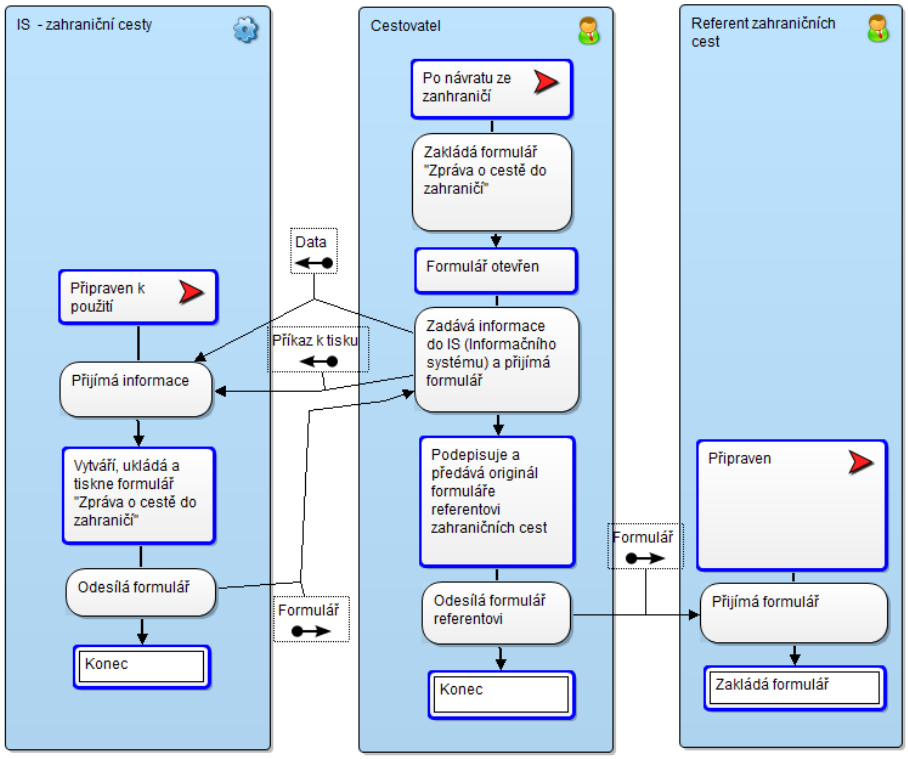
Shrnutí Výstupy z této kapitoly jsou detailněji rozpracovány v následujících kapitolách: metoda automatizované transformace modelu na text v kapitole 4.1 na straně 75 a metoda automatizované transformace modelu na model v kapitole 4.2 na straně 78.

10.2 Případové studie

10.2.1 Případová studie agendy zahraničního oddělení fakulty vysoké školy



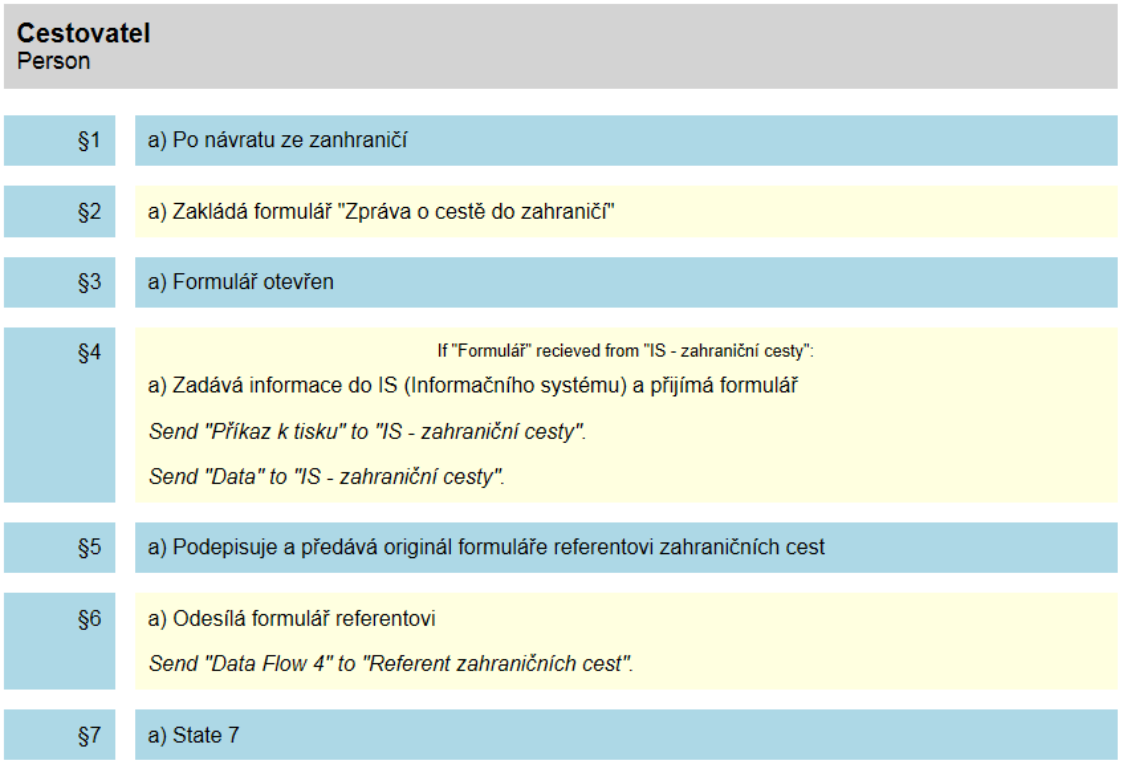
Obrázek 10.1: BPMN model zahraniční cesty, autor



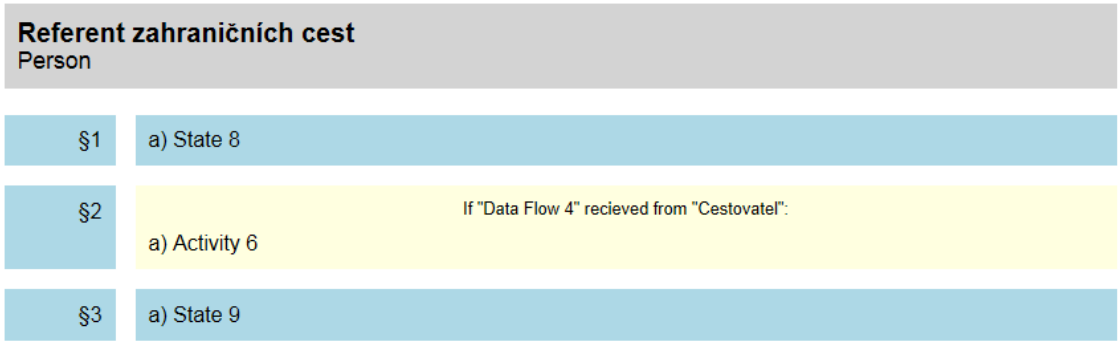
Obrázek 10.2: BORM model zahraniční cesty, autor

IS - zahraniční cesty System	
§1	a) Připraven k použití
§2	If "Příkaz k tisku" recieved from "Cestovatel": If "Data" recieved from "Cestovatel": a) Přijímá informace
§3	a) Vytváří, ukládá a tiskne formulář "Zpráva o cestě do zahraničí"
§4	a) Odesílá formulář *Send "Formulář" to "Cestovatel".*
§5	a) Konec

Obrázek 10.3: Případová studie - Operační model (manuál) pro participanta IS (Informační systém) - Zahraniční cesty, autor.

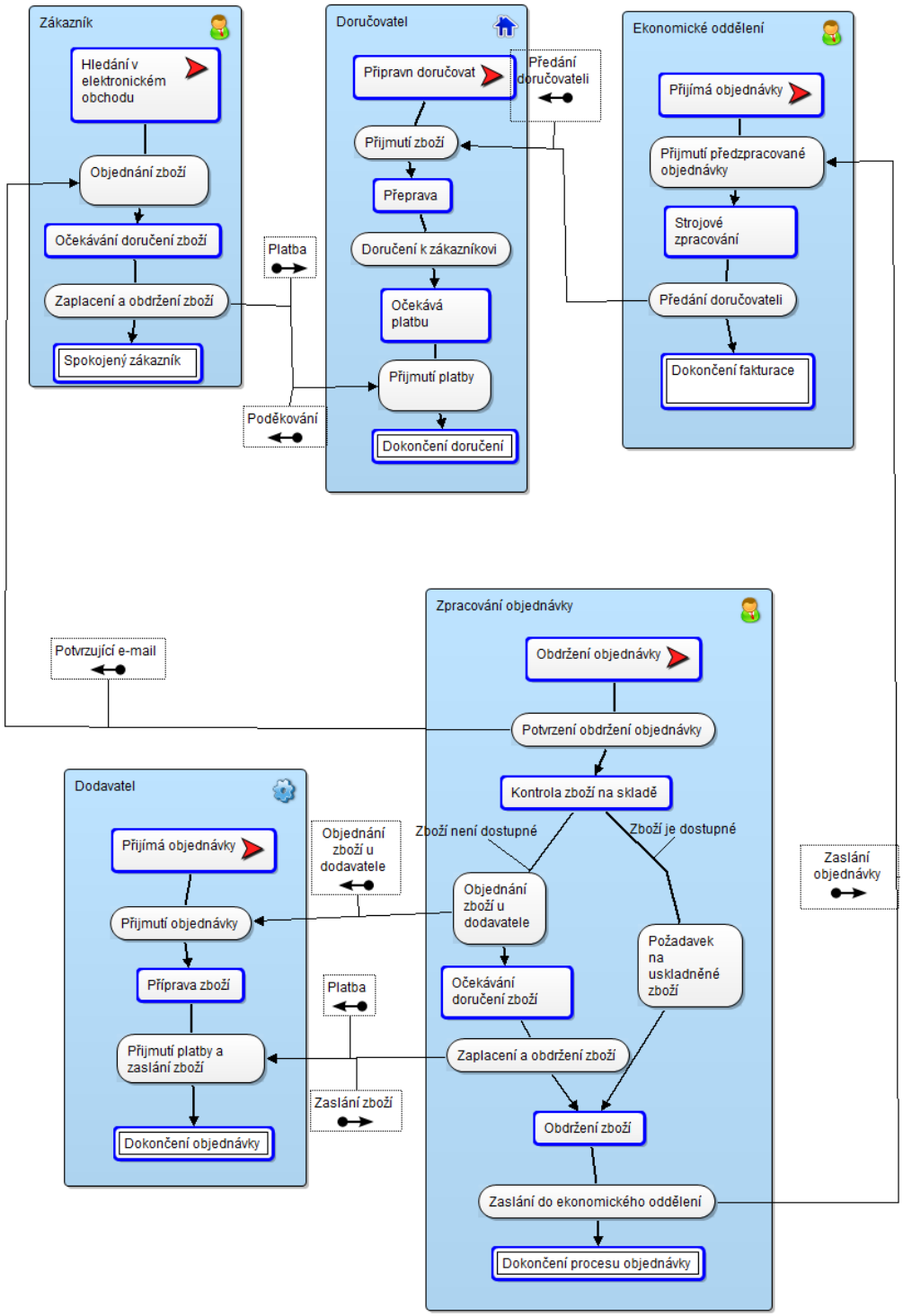


Obrázek 10.4: Případová studie - Operační model (manuál) pro participanta Cestovatel, autor.



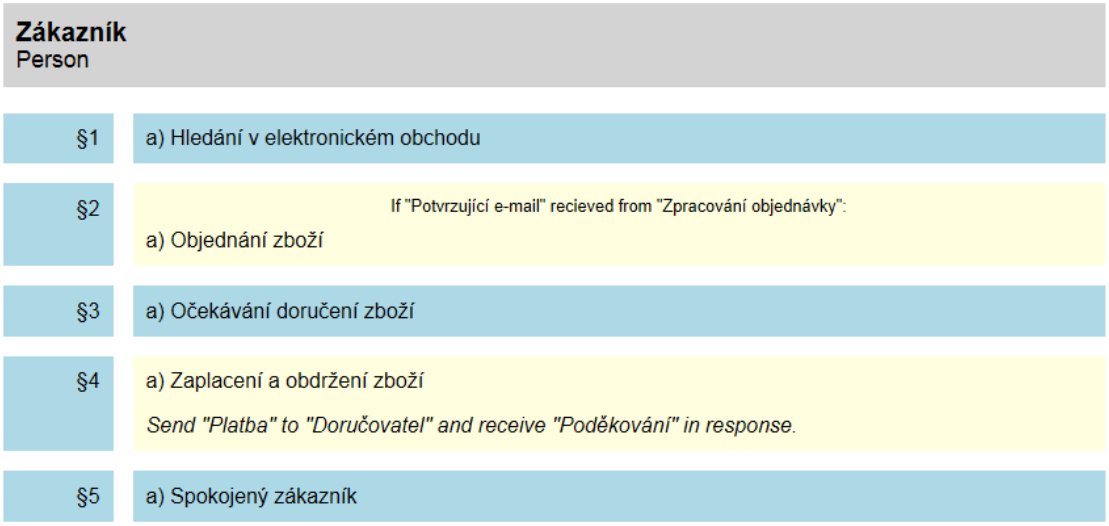
Obrázek 10.5: Případová studie - Operační model (manuál) pro participanta Referent zahraničních cest, autor.

10.2.2 Případová studie elektronického obchodu



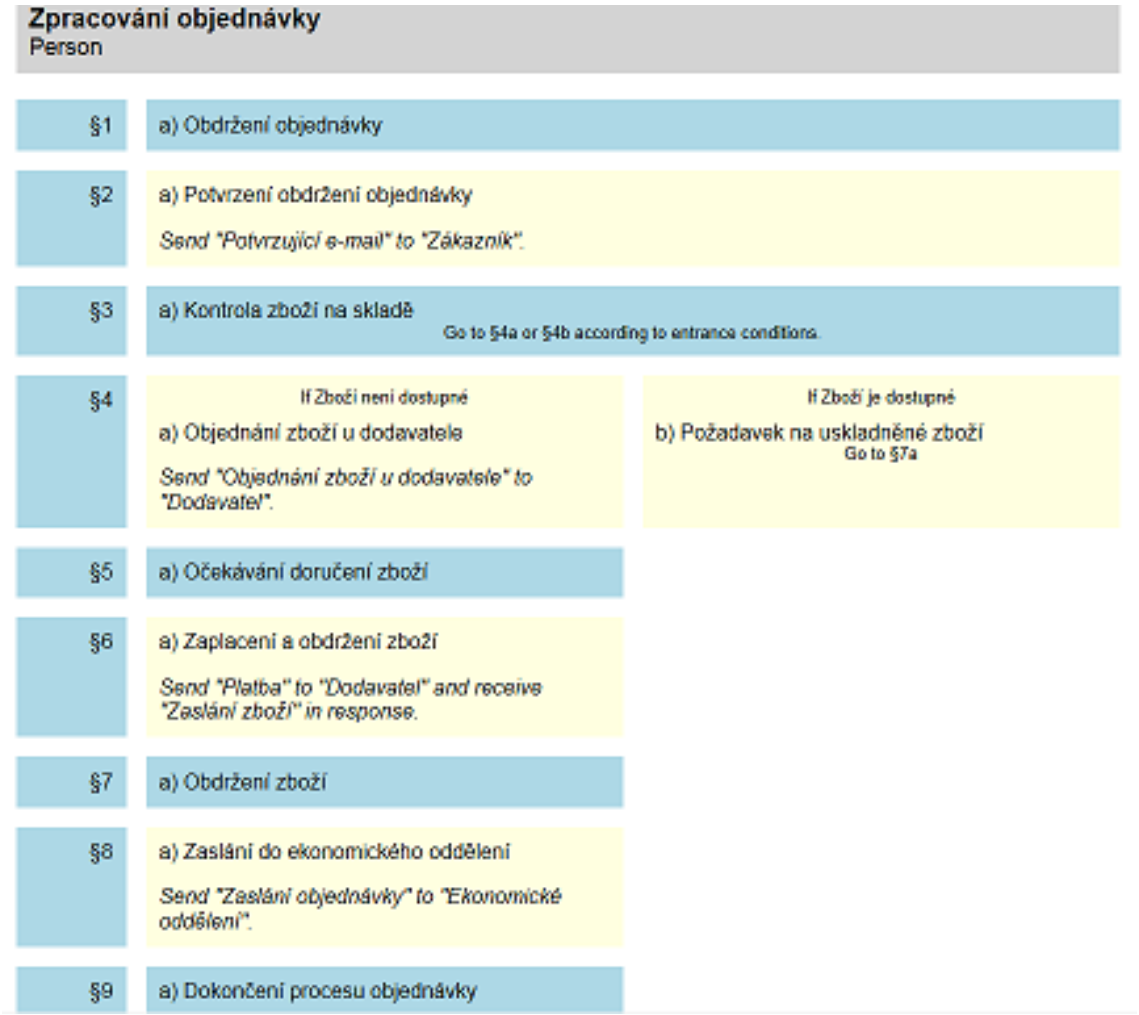
Obrázek 10.6: Případová studie - Řídící proces BORM modelu: Elektronický obchod objednání zboží. (Tůma et al., 2015)

Operační manuály v HTML jsou generovány pro každého jednotlivého participanta a zobrazeny na obrázkách 10.7 až 10.11 na stranách 147 až 149.

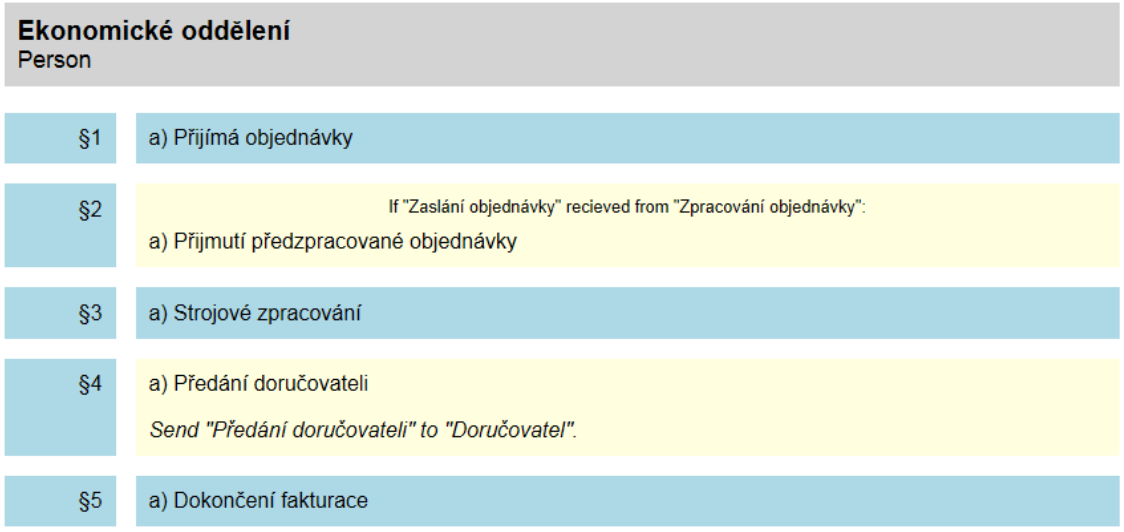


Obrázek 10.7: Případová studie - Operační model (manuál) pro participanta Zákazník. (Tůma et al., 2015)

Obrázek 10.8: Případová studie - Operační model (manuál) pro participanta Doručovatel. (Tůma et al., 2015)



Obrázek 10.9: Případová studie - Operační model (manuál) pro participanta Zpracování objednávky. (Tůma et al., 2015)

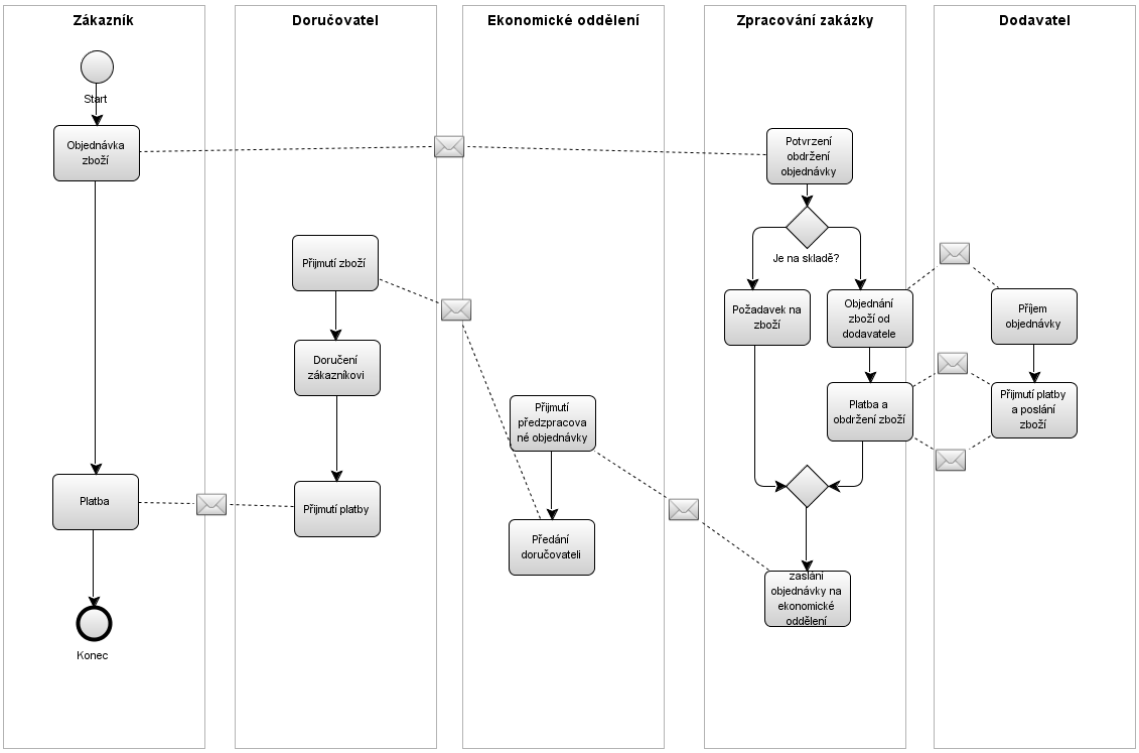


Obrázek 10.10: Případová studie - Operační model (manuál) pro participanta Ekonomické oddělení. (Tůma et al., 2015)

Dodavatel
System

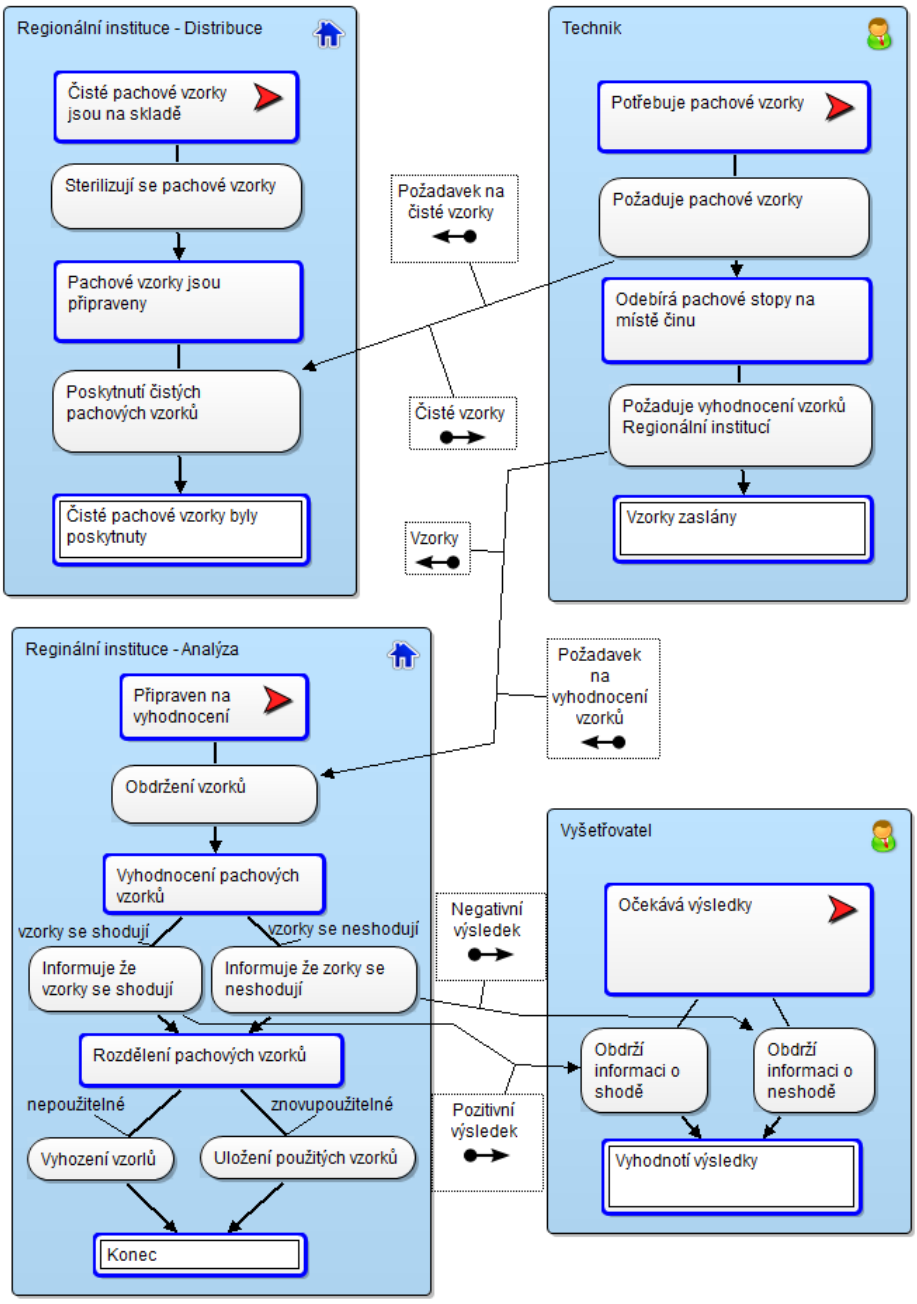
§1	a) Přijímá objednávky
§2	If "Objednání zboží u dodavatele" recieved from "Zpracování objednávky": a) Přijmutí objednávky
§3	a) Příprava zboží
§4	If "Platba" recieved from "Zpracování objednávky": a) Přijmutí platby a zaslání zboží *Send "Zaslání zboží" to "Zpracování objednávky" as response to "Platba".*
§5	a) Dokončení objednávky

Obrázek 10.11: Případová studie - Operační model (manuál) pro participanta Dodavatel. (Tůma et al., 2015)



Obrázek 10.12: Případová studie - Elektronický obchod objednání zboží v notaci BPMN, autor.

10.2.3 Případová studie metody pachové identifikace



Obrázek 10.13: Případová studie - model řídícího procesu v BORM®: Výsek z MPI (Merunka et al., 2015)

Regionální instituce - Distribuce
Organization

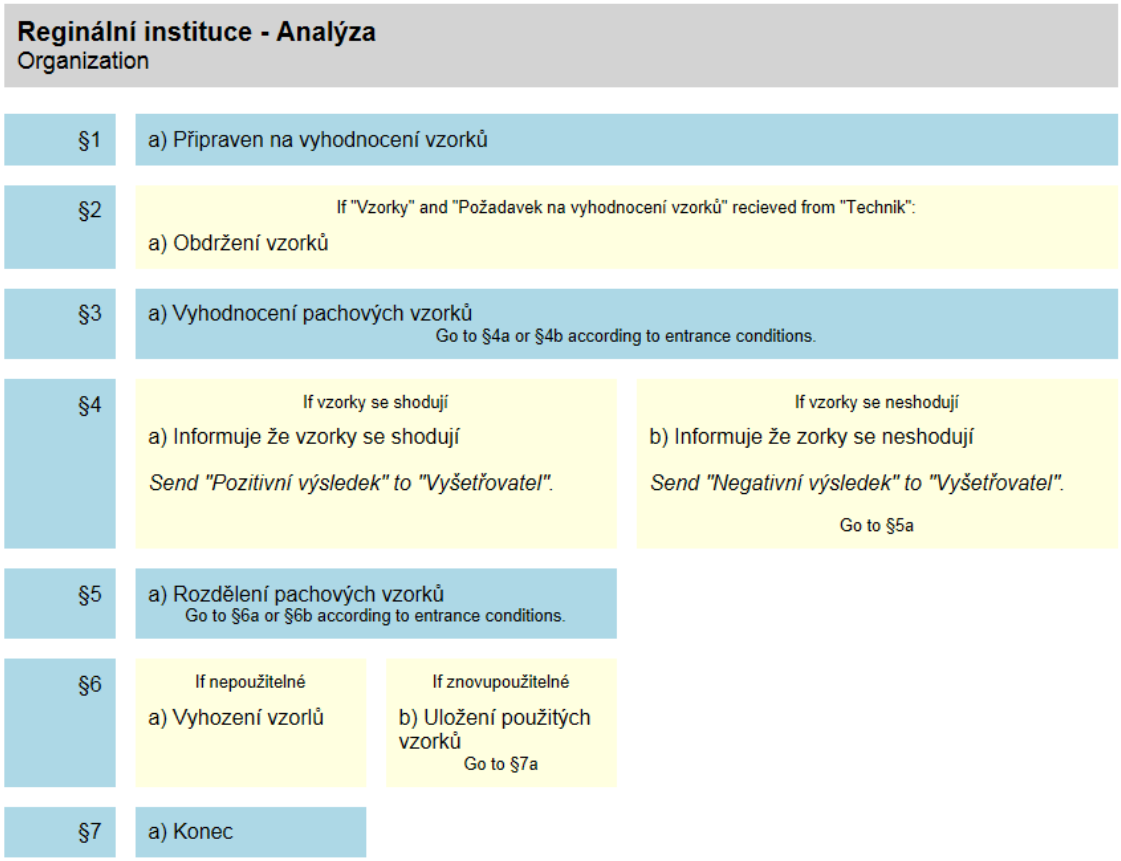
§1	a) Čisté pachové vzorky jsou na skladě
§2	a) Sterilizují se pachové vzorky
§3	a) Pachové vzorky jsou připraveny
§4	If "Požadavek na čisté vzorky" recieved from "Technik": a) Poskytnutí čistých pachových vzorků *Send "Čisté vzorky" to "Technik" as response to "Požadavek na čisté vzorky".*
§5	a) Čisté pachové vzorky byly poskytnuty

Obrázek 10.14: Operační model (manuál) pro participanta Regionální Instituce-Distribuce. (Merunka et al., 2015)

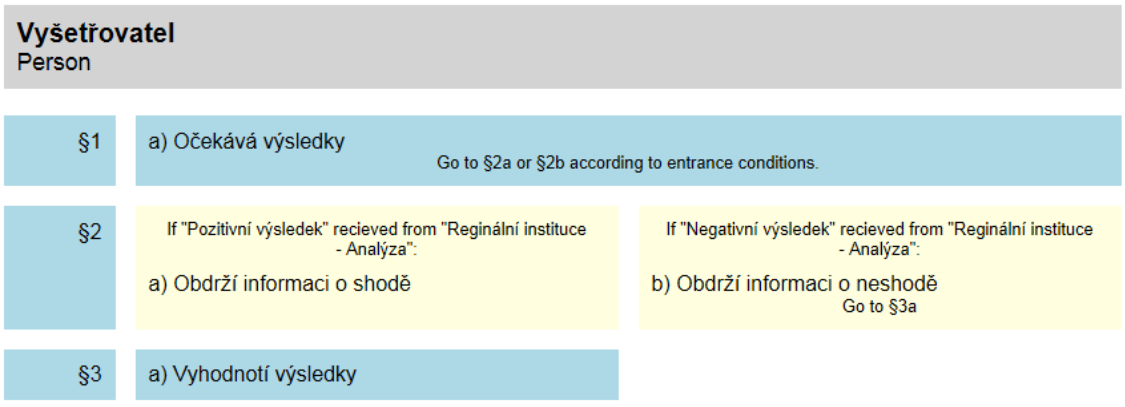
Technik
Person

§1	a) Potřebuje pachové vzorky
§2	a) Požaduje pachové vzorky *Send "Požadavek na čisté vzorky" to "Regionální instituce - Distribuce" and receive "Čisté vzorky" in response.*
§3	a) Odebírá pachové stopy na místě činu
§4	a) Požaduje vyhodnocení vzorků Regionální institucí *Send "Vzorky", "Požadavek na vyhodnocení vzorků" to "Reginální instituce - Analýza".*
§5	a) Vzorky zaslány

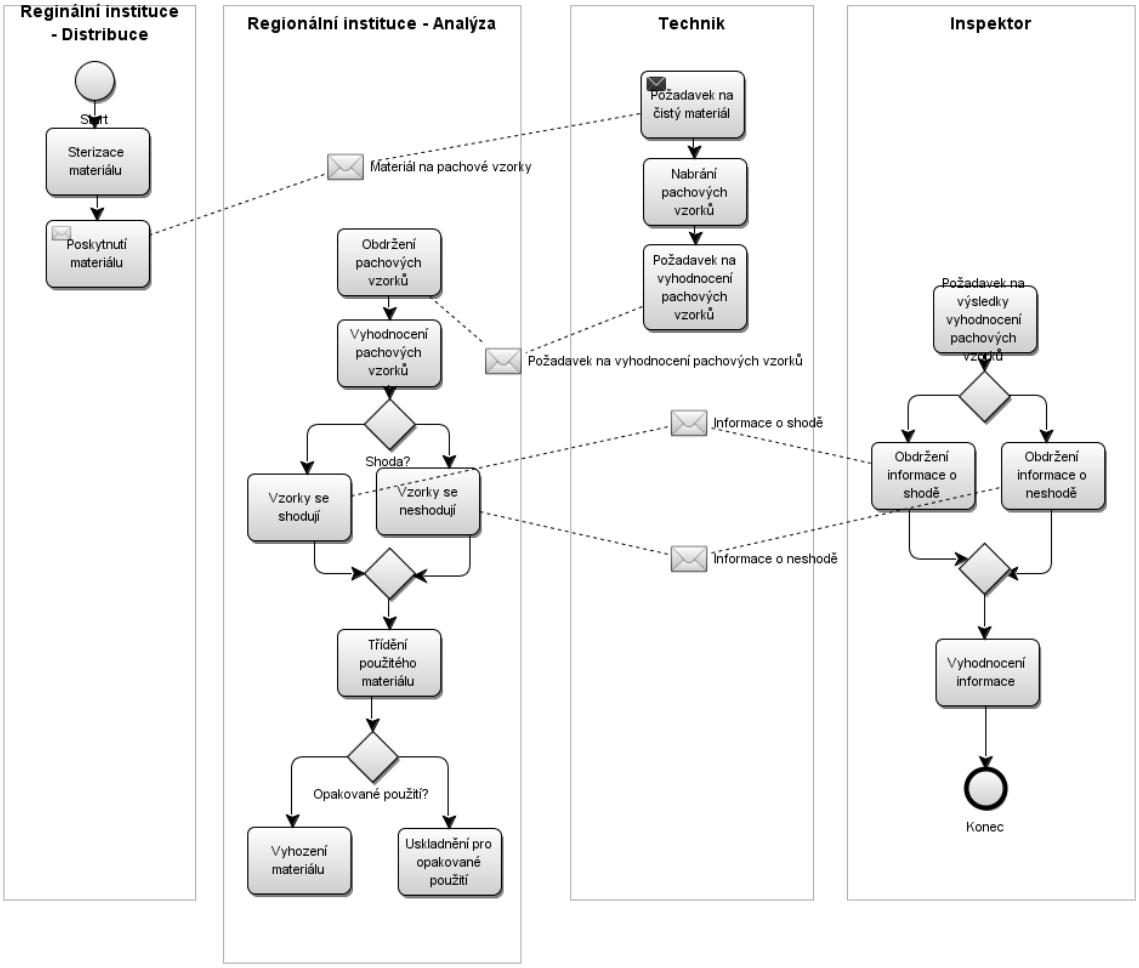
Obrázek 10.15: Operační model (manuál) pro participanta Technik. (Merunka et al., 2015)



Obrázek 10.16: Operační model (manuál) pro participanta Regionální Instituce - Analýza.(Merunka et al., 2015)

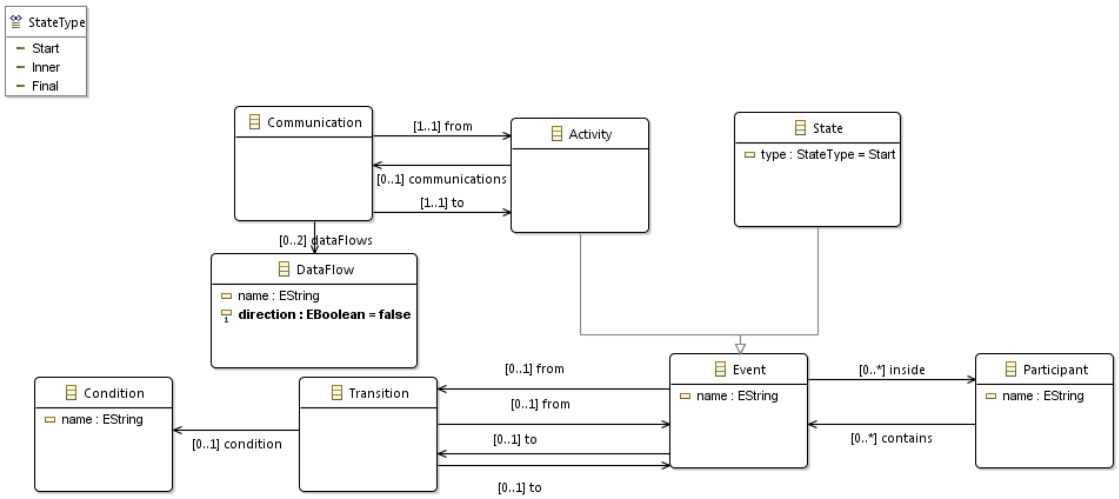


Obrázek 10.17: Operační model (manuál) pro participanta Vyšetřovatel. (Merunka et al., 2015)

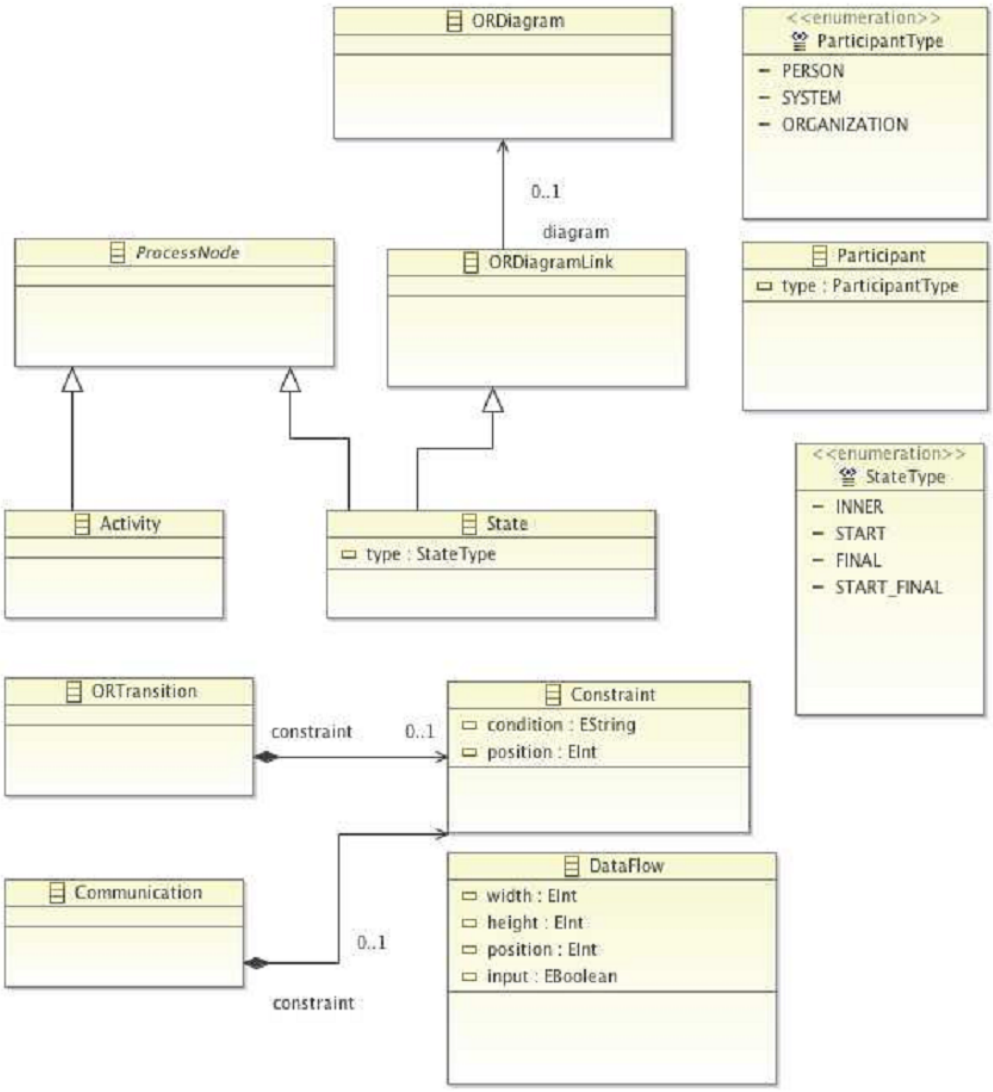


Obrázek 10.18: Případová studie MPI v notaci BPMN, autor

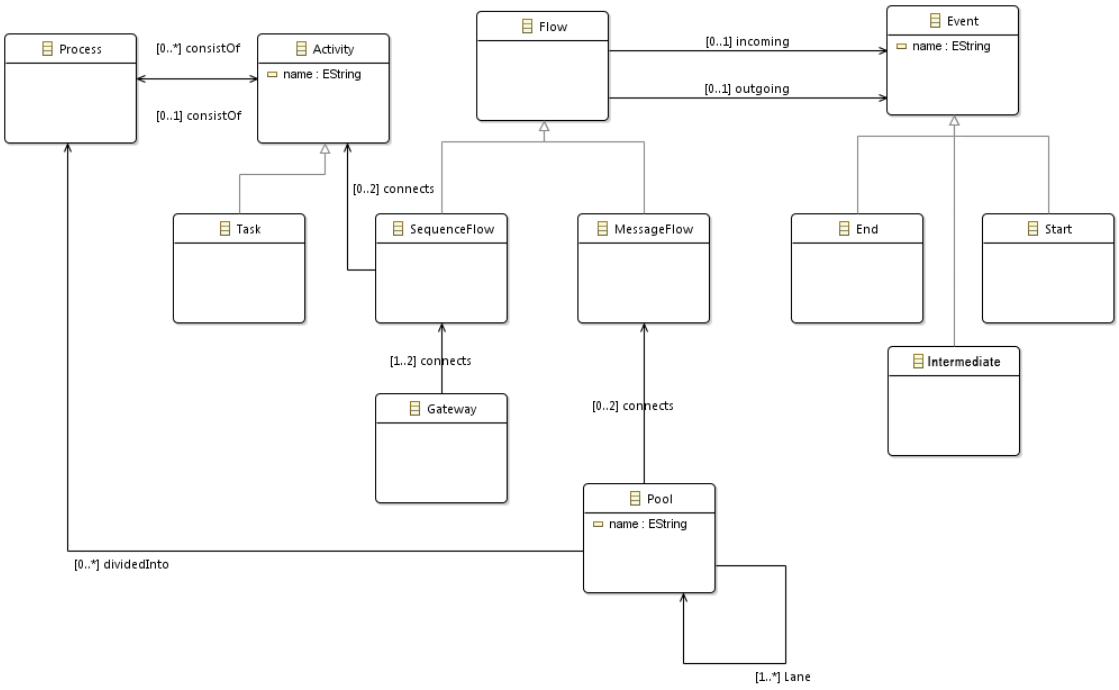
10.3 Metamodely



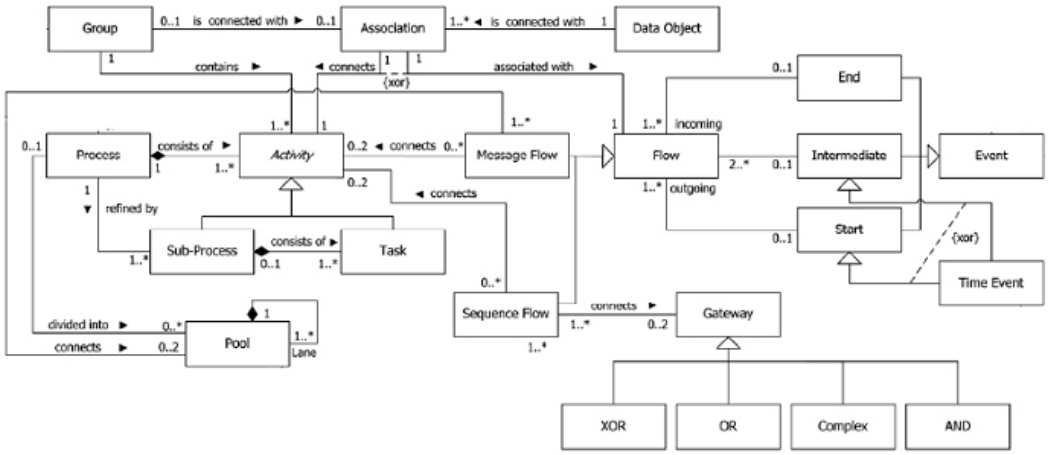
Obrázek 10.19: Metamodel ORDiagram vytvořeno autorem



Obrázek 10.20: BORM ORD metamodel (Moravec, 2014)



Obrázek 10.21: Metamodel PSDiagram BPMN vytvořeno autorem



Obrázek 10.22: BPMN metamodel (Korherr & List, 2007)