Česká zemědělská univerzita v Praze
Provozně ekonomická fakulta

Orchestrace a choreografie procesů v BORM

Disertační práce z oboru Informační Management

Ing. Josef Moravec

Školitel: doc. Ing. Prokop Toman, CSc.

Chtěl bych na tomto místě poděkovat svému vedoucímu doc. Ing. Prokopovi Tomanovi, CSc. za veškeré rady, připomínky a odborné vedení při psaní této práce.

Dále bych rád poděkoval mým kolegům z Katedry informačního inženýrství, kteří mi byly vždy ochotni pomoci a poradit a v nichž jsem našel nejen odborníky ve svém oboru, ale také přátele.

V neposlední řadě bych na tomto místě rád poděkoval mým rodičům a všem mým blízkým, kteří mi vždy byli velkou oporou.

Josef Moravec

Orchestrace a choreografie procesů v BORM

Klíčová slova

Orchestrace, choreografie, BORM, modelování procesů, konečné automaty

Abstrakt

Práce se zabývá problematikou řízení komunikace mezi interními a externími účastníky obchodních procesů s využitím metody BORM. Cílem práce je jednoznačně formálně popsat procesní diagramy v BORM pomocí teorie konečných automatů, tak aby bylo možné tímto způsobem zaznamenat nejen řídící toky, ale také datové toky. Dílčím cílem práce je rozšířit metodu BORM případně její části tak, aby bylo možné lépe a efektivněji modelovat choreografii v rámci procesů.

Business process orchestration and choreography using BORM method

Keyword

Orchestration, choreography, BORM, process modeling, finite state machines

Abstract

The thesis deals with the communication between internal and external users of business processes using BORM method. The aim of this work is to formally and clearly describe the process diagrams in BORM using the theory of finite state machines, in order to describe not just control flows but also the data flows. Furthermore the work suggests extension of the BORM method so that it can be used for better and more efficient modeling of the choreography inside business processes.

Obsah

1 Úvod

Business Process Management (BPM) je komplexní manažerský přístup zahrnující metody, techniky a nástroje pro podporu analýzy, návrhu, zavádění a správy provozních obchodních procesů (van der Aalst, a další, 2003). Během posledních dvaceti let došlo k výraznému zvýšení zájmu o oblast řízení podnikových procesů. Díky veliké poptávce po BPM existuje pro řízení podnikových procesů mnoho metod, technik a nástrojů.

Stále více společností investuje za účelem zvyšování efektivity práce a snižování provozních nákladů nemalé prostředky do analýzy, reingeneeringu nebo auditu procesů probíhajících ve firmě. Důležitým předpokladem pro efektivní fungování podniku jsou dobře zdokumentované procesy, které se snadno udržují a jednoduše provádějí. Dobře navržené a spravované procesy snižují jak časovou, tak i finanční náročnost na provoz (Brocke a Rosemann, 2010). Jednodušší procesy jsou lépe proveditelné a často spolehlivější. Rozmach distribuovaných technologií a potažmo servisně orientovaných architektur umožnil snadnější interakce napříč společnostmi. Internet již dávno není pouze prostorem pro sběr a prezentaci informací. Rozvoj webových služeb znamenal posun internetu od úložiště informací k nástroji pro distribuované výpočty.

Podle zkušeností autora práce může využití webových služeb jako účastníků procesu zjednodušit návrh procesu tím, že se eliminuje nutnost realizovat úkol, který služba zabezpečuje, pomocí do nedávné doby běžných konstrukcí. Běžnou konstrukcí je myšleno například objednávání zboží zaměstnancem jedné společnosti prostřednictvím sytému společnosti jiné. Objednatel může pověřit zaměstnance, který bude prostřednictvím systému poskytovaným dodavatelem zboží objednávat. Takový koncept má svá úskalí při plánování procesu. Nahradíme-li pověřeného zaměstnance a systém dodavatele webovými službami, lze jednak mnohdy zjednodušit návrh procesu a také je možné do logiky rozhodování procesů začlenit parametry jejich vzájemné komunikace. Využití webových služeb může vést ke zjednodušení procesů a snížení provozních nákladů na proces alokovaných. Na druhé straně je při návrhu procesů obsahujících jak reálné uživatele (lidi), tak i webové služby, nutné ošetřit různá chování webových služeb na konkrétní vstupy a pochopitelně také výjimky, aby nedocházelo k zacyklení nebo uváznutí procesu. Součástí Business Procees Managementu je řízení průběhu procesů, ať již jsou jejich účastníci reální uživatelé nebo webové služby. Problematikou užití webových služeb jako účastníků procesů se zabývá orchestrace a choreografie.

Orchestrace nebo také někdy instrumentace, je název odvozený z hudební teorie, který vyjadřuje techniku hudební skladby mající za cíl optimální využití rozličného počtu hudebních nástrojů za účelem dosažení požadovaného efektu při reprodukci hudební skladby. Orchestr je řízen dirigentem. Přeneseno na problematiku procesů je možné orchestraci chápat jako koordinaci procesů a jejich účastníků, kteří jsou reprezentováni pomocí webových služeb. Orchestrace popisuje pořadí vykonávání interakcí webových služeb a může zahrnovat jak interní, tak externí webové služby. Při orchestraci je proces vždy řízen jednou stranou, kterou označujeme jako orchestrátora. Orchestrace je podrobně popsána v kapitole 8.2.

Obdobnou analogií lze popsat choreografii, jejíž definici a podrobnému popisu je věnována kapitola 8.1. Choreografie je činnost spočívající ve vytvoření koncepce, kompozice a řízení tanečního díla. Jednotliví tanečníci tančí podle předem daného scénáře, aniž by byli řízeni. Přeneseme-li opět koncept taneční choreografie na procesy, lze říci, že choreografie popisuje spolupráci a vzájemnou interakci účastníků procesu nebo webových služeb za účelem dosažení společného cíle. Účastníci procesu popisují svoji roli v rámci choreografie. Na rozdíl od orchestrace zde není určen řídicí element a interakce se popisují z globálního pohledu.

2 Cíle

Práce se zabývá problematikou řízení komunikace mezi interními a externími účastníky obchodních procesů s využitím metody BORM. Jednotliví participanti mohou být reprezentováni jak webovými službami, tak i reálnými uživateli. Obecně je pro takový způsob řízení komunikace nazýván orchestrací, respektive choreografií procesů.

Hlavním cílem této práce je jednoznačně formálně popsat procesní diagramy v BORM pomocí teorie konečných automatů, tak aby bylo možné tímto způsobem zaznamenat nejen řídící toky, ale také datové toky.

Aby mohlo být tohoto cíle dosaženo, je nutné splnit následující dílčí cíle:

- Analyzovat literární zdroje zabývající se problematikou procesního modelování, teorie konečných automatů a webových služeb.
- Formálně popsat procesní digram v BORM jako konečný automat a nalézt analogický koncept v oblasti konečných automatů, tak aby tento koncept bylo možné využít pro záznam komplexního chování procesu.

Dílčím cílem práce je rozšířit metodu BORM případně její části tak, aby bylo možné lépe a efektivněji modelovat choreografii v rámci procesů.

Dalším dílčím cílem práce je na základě komparace vybraných jazyků pro popis webových služeb a jazyku, který je použit pro definicí procesů v nástroji IZMAN Case navrhnout změny v jazyce, který je aktuálně pro záznam procesů používán s ohledem na možnost definovat orchestraci a choreografii.

3 Metodika

První část práce bude obsahovat analýzu literárních pramenů z oblasti webových služeb, mapování obchodních procesů a teorie konečných automatů. Se zvláštním zaměřením na metodu BORM budou popsány vybrané metody pro modelování obchodních procesů, na kterých budou demonstrovány možnosti modelování orchestrace a choreografie. Jelikož se autor práce nejčastěji setkal s pojmy orchestrace a choreografie v souvislosti s webovými službami, budou představeny základní přístupy pro popis webových služeb, včetně jejich klíčových parametrů umožňující zápis choreografie a orchestrace.

Na základě získaných poznatků a přehledu přístupů k dané problematice bude sestaven formální popis procesního diagramu. Dále budou aplikovány přístupy z teorie konečných automatů tak, aby tento popis plně reflektoval potřeby vyplývající z grafické podoby procesních diagramu v BORM.

Dále budou popsány možnosti zachycení choreografie procesů v BORM a bude navrženo rozšíření této metody tak, aby bylo možné choreografii lépe a efektivněji modelovat.

Na základě komparace vybraných jazyků pro popis webových služeb s jazykem používaným pro popis procesů v nástroji IZMAN Case budou definována jeho slabá místa tohoto jazyka s ohledem na zápis orchestrace a choreografie. Bude-li to nezbytné, bude navrženo jeho rozšíření, tak aby tento byl srovnatelný s jazyky běžně k obdobnému účelu používanými.

4 Vybrané nástroje pro modelování procesů, orchestrace a choreografie

Podíváme-li se hluboko do historie, setkáme se s procesy v každé etapě lidského vývoje. Od jeskyních lidí, stavitelů pyramid přes období renesance, jehož výrazným představitelem je Leonardo da Vinci, až do dnešní moderní doby se lidé vždy potýkali s nutností vykonávat určité činnosti(procesy) přičemž si uvědomovali, že v rámci konkrétního procesu zosobňují určitou roli. Velké osobnosti často předběhli svoji dobu a jelikož mezi jejich současníky nebyl nikdo, kdo by byl schopen uvažovaní na stejné úrovni jako oni, zastávali často dvojjediné role. Vezměme například již zmiňovaného Leonarda da Vinci, který byl nejen vynálezcem, ale také konstruktérem. Ačkoli bychom našli mnoho zástupců výjimečných osobností, které byly schopny zastávat více rolí, většina lidí není schopna takové komplexnosti. V lidských silách, ať už mentálních nebo fyzických, není možné pojmout veškeré potřebné vědění či um pro řešení komplexního problému a je proto nezbytná specializace. Velká složitost problému je často těžko uchopitelná a nad možnosti naší mentální kapacity. Jak se tedy vypořádat s problémem, který je jako celek nad možnosti chápání jedné osoby?

Jako logické se jeví rozdělit složitý problém na více méně složitých problémů. Komplexním problémem je z našeho pohledu návrh a implementace informačního systému. Architektura budov je dobrou paralelou k budování informačních systémů. Obdobně jako architekt přenese svoji vizi stavby do podoby plánů a výkresů, přenáší architekt informačních systémů požadavky uživatelů do podoby procesních diagramů a datových modelů. Stejně jako v případě stavitelů, kdy jeden člověk dokázal stavbu nejen navrhnout, ale také postavit nebylo v minulosti výjimkou, že jeden člověk nejen navrhoval ale také implementoval informační systém. Jak se postupně prohlubovaly znalosti lidí v oblasti statiky a nauky o matriálech, oddělila se role architekta a stavitele. Architekt navrhuje stavbu tak, aby co nejlépe vyhovovala potřebám uživatele. Naproti tomu stavitel realizuje záměry architekta přičemž musí zajistit, aby výsledek splňoval kvalitativní požadavky, ale také cenu, kterou si stanovil uživatel a ze které při návrhu vycházel i architekt. Podobně jako v případě stavby budov se také v případě budování informačních systémů vyhranili dvě profese. Návrhář architektury informačního sytému a ten, kdo bude tento návrh implantovat by měli vzájemně spolupracovat stejně jako architekt a stavitel. Jak uvádí (Polák, a další, 2003) dochází v případě realizace informačních systémů mnohem

častěji k zásadním změnám projektu než u výstavby budov. Důvodem bývá špatná architektura nebo lépe řečeno návrh informačního systému, který neodpovídá požadavkům uživatele.

Počítače již dávno nejsou nástrojem několika vyvolených, kteří byli nuceni naplno a velmi pečlivě využívat jejich velmi omezené prostředky. Rozvoj informačních technologií znamenal také rozvoj nástrojů a technik pro tvorbu informačních systémů. Přestože existuje mnoho CASE nástrojů, které usnadňují návrh informačních systémů, jde pouze o nástroje. Bez potřebných znalostí není možné ani se sebelepším CASE nástrojem vybudovat systém, který by odpovídal požadavkům uživatele (Shlaer a Mellor, 1992). Nicméně ani dokonalé znalosti a zvládnutí práce s CASE nástrojem nezaručují dobrý výsledek. Klíčovým problémem je pochopení, jaký má výsledný systém být a co je pro jeho uživatele důležité. Je zapotřebí pochopit problém uživatele, pochopit jeho procesy a vytvořit most mezi uživatelem a realizátorem informačního sytému (Polák, a další, 2003). Úkolem architektů informačních systémů je překlenout mezeru mezi světem uživatelů a světem těch, kteří budou systém realizovat. Architekti informačních systémů využívají pro převedení uživatelského pohledu na problém do podoby pochopitelné pro realizátory metody a pracovní postupy, jejichž vybranými zástupci se v této práci budeme zabývat.

Na pomezí architekta a realizátora informačního systému stojí systémový integrátor. Je zde celá řada definic pojmu systémový integrátor. Podle šetření, které provedla Česká společnost pro systémovou integraci (Habáň a Sodomka, 2004) je systémový integrátor společnost, která má přímý vztah k zákazníkům a je vždy spoluřešitelem IS/IT projektů u zákaznických organizací. Podle Poláka (Polák, a další, 2003) je systémový integrátor osoba nebo firma, která umí účelně vyhledat, provázat a propojit jednotlivé komponenty informačního systému do jednoho celku tak, aby uspokojil potřeby uživatele a podpořil jeho procesy. Systémový integrátor by tedy měl zastat roli architekta a navrhnout podobu budoucího informačního systému. Dále by měl být schopen tento svůj návrh realizovat buď za použití již hotových komponent a stávajících částí informačního systému, a nebo vytvořením komponent nových.

V osmdesátých letech dvacátého století byl kladen značný důraz na TQM (Total Quality Management), což je filozofie řízení pro trvalé zvyšování kvality produktů a procesů. Začátkem devadesátých let byla tato filozofie následována BPR (Business Process Reingeneering), který měl bohatou historii s mnoha velkými úspěchy a pochopitelně i neúspěchy. BPR se zabývá analýzou, návrhem a přepracováním pracovních postupů a procesů v rámci organizace za účelem zvýšení jejich efektivity (Jeston a Nelis, 2008).

Počátkem jednadvacátého století se objevil nový směr, který vychází z myšlenky BPR. Business Process Management (BPM) zahrnuje metody, techniky a nástroje pro podporu návrhu, řízení a analýzu obchodních procesů (van der Aalst, a další, 2003). Někteří autoři jako (Jeston a Nelis, 2008), nepředkládají konkrétní definici BPM, ale popisují ho jako interdisciplinární přístup zahrnující informační technologie a management, jehož předmětem zájmu jsou organizace, lidé a informační technologie tvořící obchodní procesy. Podobně jako v případě BPR je cílem nejen procesy řídit, ale i měřit a případně zvyšovat jejich efektivitu.

Obchodní proces[1], nebo též podnikový proces, někdy také označovaný jako obchodní případ, vymezuje (Ould, 2005) jako ucelenou množinu aktivit, které jsou vykonávány vzájemně spolupracujícími účastníky za účelem dosažení určitého cíle. Obdobným způsobem vymezují pojem obchodního procesu (Smith a Fingar, 2007), kteří uvádějí, že obchodní proces je ucelená a dynamicky koordinovaná množina spolupracujících a transakčně provázaných aktivit, které přinášejí hodnotu pro zákazníky. Obecně lze tedy říci, že obchodní proces je posloupnost vzájemně propojených aktivit a stavů mající výchozí a konečný stav. Výsledkem procesu je přidaná hodnota pro zákazníka procesu. Během vykonávání procesu dochází k transformaci vstupní informace na informaci výstupní.

Modely obchodních procesů by měly mít formální základ, protože formální modely umožňují problém jednoznačně popsat a lépe analyzovat. Procesní model by měl být pochopitelný pro všechny zúčastněné strany, což by mělo být dosaženo přehlednou grafickou reprezentací procesu. Vlastníci, účastníci, vývojáři i analytici by měli být schopni procesnímu modelu přiřadit stejný, jednoznačný význam, který neumožňuje alternativní výklad (Jeston a Nelis, 2008).

4.1 Techniky modelování obchodních procesů

Ačkoli existuje celá řada rozšířených i méně používaných přístupů a technik pro modelování obchodních procesů, budeme se zabývat pouze vybranými zástupci. Následující přehled nejpoužívanějších technik a přístupů pro modelování obchodních procesů předkládá (Giaglis, 2001).

Flowcharting. Flowcharting, nebo též vývojové diagramy, jsou jednou z prvních modelovacích technik, která vznikla v šedesátých letech dvacátého století. Výhody vývojových diagramů spočívají ve schopnosti ukázat celkovou strukturu procesu, sledovat toky informací, popisovat média, na kterých jsou informace

[1] Označení obchodní proces je překladem anglického názvu Business Process, lze se také setkat s označením podnikový proces nebo obchodní případ

předávány, a ve schopnosti zobrazit klíčová místa a rozhodovací body procesu. Původně byly vývojové diagramy používány pro zobrazení algoritmů a logiky počítačových programů, ale vzhledem k jejich obecné povaze je lze aplikovat i na další oblasti včetně modelování obchodních procesů. Jedná se spíše o doplňkovou techniku pro popis procesu.

Petriho sítě. Petriho sítě se nejčastěji používají pro modelování a synchronizaci paralelních výpočtů nebo modelování výrobních procesů (montážní linky, apod.). Jejich použití se neomezuje pouze na aplikace v technické oblasti. V posledních letech se také používají k popisu, analýze, simulaci a modelování obchodních procesů (Girault a Valk, 2003). Petriho síť je orientovaný, ohodnocený, bipartitní graf, který se skládá z míst, přechodů a hran. Základní Petriho sítě jsou matematicko-grafické reprezentace procesů, zaměřené na podporu analýzy struktury a dynamiky chování zejména paralelních systémů.

Teorie automatů. Základy teorie automatů položili McCuloch a Pittsov, kteří roku 1943 navrhli model abstraktního zařízení s konečným počtem dosažitelných stavů. Jedním z nejstarších a nejznámějších způsobů modelování chování systémů uvádějí (Gill, 1962) a (Wright, 2005) stavové automaty. Konečné automaty dovolují zachytit a popsat stav systému nebo procesu v konkrétním místě a čase. Chování celého systému je možné odvodit na základě jeho stavů. Použití této techniky není omezeno volbou konkrétního modelovacího nástroje a je nezávislé na implementaci. Jelikož je hlavním cílem této práce popsat procesní diagramy v BORM pomocí teorie konečných automatů, bude této problematice věnována samostatná kapitola.

Data flow diagramy (DFD). DFD jsou technikou pro grafický záznam datových toků mezi externími entitami, interními kroky zpracování a data-store elementy v rámci obchodního procesu. DFD se používají pro popis systému se zaměřením na toky dat do systému, v rámci systému a vně systému. V tomto ohledu jsou DFD srovnatelné s vývojovými diagramy. Liší se od nich v podstatě pouze v zaměření. DFD se zaměřují na data, zatímco vývojové diagramy na aktivity a kontrolu řízení. DFD byly široce používány pro datové modelování a staly se standardní notací pro analýzu a návrh systémů, nicméně mají řadu omezení. DFD se převážně zaměřují na data a neposkytují žádné konstrukty pro popis work flow, lidí, událostí a dalších prvků, z nich se podnikové procesy skládají. Nedefinují počáteční ani koncové body procesu. Jedná se o statickou reprezentaci funkcionalit systému, která zahrnuje manipulaci s daty. Z toho důvodu je obtížné jejich použití pro analýzu nebo rozhodování o řízení.

State-transition diagramy (STD). State-transition diagramy vycházejí z analýzy a návrhu systémů zpracovávajících úlohy v reálném čase. ST diagramy se pokoušejí překonat omezení vyplývající ze statického charakteru DFD tím, že poskytují explicitní informace o časovém sledu událostí. Notace state-transition diagramů je velmi jednoduchá. Skládá se pouze z obdélníků, které představují stavy, a šipek, které představují změny stavů (přechody). Ačkoli se podařilo ST diagramům překonat některá omezení data flow diagramů, stále se soustřeďují primárně na data a opomíjí work flow, řízení, rozhodování a další.

Unified Modelling Language (UML). V roce 1997 přijala OMG (Object management Group) jazyk UML jako standard pro specifikaci, konstrukci, vizualizaci a dokumentaci softwarových systémů. UML využívá širokou škálu diagramů zahrnujících:

- *Use Case* diagramy pro zobrazení struktury systému z pohledu uživatele.
- *Class diagramy* pro zachycení statického pohledu na systém, znázornění objektů v systému a jejich vztahů.
- Diagramy po popis chování, což jsou již zmiňovaný *Use Case diagram*, dále *statechart*, *activity*, *state machine* a *interaction* diagramy.
- Diagramy po popis struktury, mezi něž patří například *Class diagramy*, *Component diagramy* a další.

Navzdory velkému množství dlouho existujících jazyků a prostředků pro popis softwaru, většina z nich je spjata s konkrétními způsoby analýzy a návrhu. Tato zdánlivá rozmanitost může být zdrojem problémů interoperability mezi jazyky. UML se snaží tuto mezeru zaplnit svou univerzálností, tedy tím, že pokrývá celý proces návrhu od modelů obchodních procesů až po databázové schéma a softwarové komponenty. V rámci této práce se budeme nadále věnovat pouze notaci BPMN a metodě BORM.

4.2 Business Process Modeling Notation

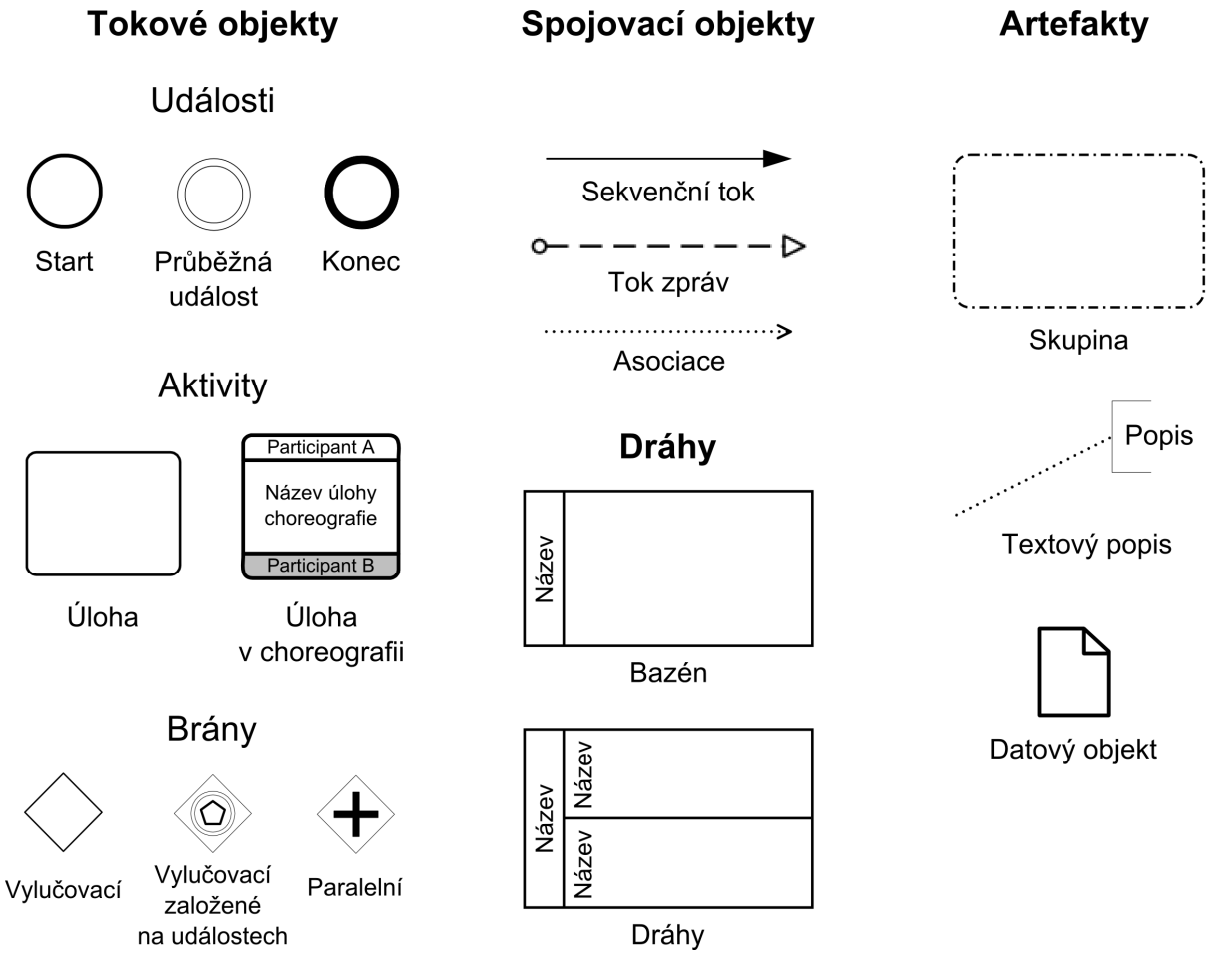
Během několika posledních let došlo ke značnému rozvoji vykonávacích jazyků založených na XML pro řízení podnikových procesů. Jazyky jako BPEL4WS poskytují formální nástroj pro definici obchodních procesů. Klíčovým prvkem těchto jazyků je to, že jsou optimalizovány pro zajištění provozu a spolupráce BPM systémů, následkem čehož jsou tyto jazyky méně vhodné pro přímé použití návrháři procesů. Vzhledem k povaze jazyků, jako je BPEL4WS, je možné zaznamenat proces sice potenciálně komplexní, ale v nesouvislém a neintuitivním formátu. Pro programátory a informační systémy je srozumitelný, ale je těžko pochopitelný pro procesní analytiky a manažery, kteří mají za úkol proces

modelovat, sledovat a řídit. Pro manažery a analytiky je pohodlné a jednodušší zaznamenat proces graficky, pomocí vývojových či flow-chart diagramů. Rozdíl mezi způsobem počátečního návrhu podnikových procesů a jazyky, jako BPEL4WS, používaných pro vykonávání procesů, tvoří mezeru, kterou je třeba překlenout. Mezera musí být odstraněna pomocí formálního mechanizmu, který umožní mapovat grafický zápis procesu na odpovídající BPM vykonávací jazyk.

BPMN dovoluje také mapovat diagramy na vykonávací jazyky, jako BPEL4WS, čímž poskytuje standardní mechanizmus pro vizualizaci procesů definovaných ve vykonávacím jazyce. Rozšiřuje možnosti tradičních notací pro zápis obchodních procesů o možnost modelovat komplexní B2B procesy, tedy veřejné a soukromé procesy a choreografie. Dále dovoluje definovat způsoby zpracování výjimek, transakce a kompenzace výpadků komunikace.

Jedním z důvodů pro vývoj BPMN bylo vytvoření mechanizmu pro tvorbu modelů podnikových procesů, který bude na jedné straně jednoduchý a srozumitelný a na straně druhé bude schopen zachytit i velmi složité procesy. Splnění těchto, do značné míry protichůdných, požadavků bylo dosaženo rozčleněním grafických prvků, z nichž se proces skládá, do pěti základních kategorií. Malá množina základních grafických prvků dovoluje vlastníkům a účastníkům procesu proces snadno pochopit a dobře se v něm orientovat. Základní kategorie obsahují další, již specifické grafické prvky daného druhu, které dovolují vytvářet detailnější modely procesů. Jelikož grafické prvky v dané kategorii vycházejí ze stejného základu, nedochází při jejich použití k výrazné změně grafické podoby procesu a tedy ani čitelnost a pochopitelnost ze strany vlastníků, účastníků a koncových uživatelů procesu není omezena. Specifikace BPMN (OMG BPMN, 2011) uvádí těchto pět základních kategorií grafických prvků procesu:

- Tokové objekty
- Data
- Spojovací objekty
- Dráhy (Swimlanes)
- Artefakty



Obr. 4.1. Základní grafické elementy notace BPMN – vlastní zpracování autora podle (OMG BPMN, 2011)

4.2.1 Tokové objekty

Základní grafické prvky, které definují chování obchodního procesu, jsou tokové objekty. Jednotlivé tokové objekty lze navzájem propojit pomocí spojovacích objektů. Rozlišujeme tři druhy tokových objektů – *události*, *aktivity* a *gateways*, neboli brány.

Události. Událost, jak ji definuje například (Chandy, 2006), je významná změna stavu. Čistě intuitivně lze za událost považovat cokoli, co se stane v průběhu procesu nebo choreografie. Události mají vliv na tok uvnitř modelu procesu. Obvykle označují nějakou příčinu nebo důsledek. Zakreslujeme je symbolem kružnice. Pomocí doplnění značek uvnitř symbolu rozlišujeme rozdílné příčiny (spouštěče) a výsledky událostí. Existují tři základní typy událostí – *start*, *průběžná událost* a *konec*. Rozlišujeme je podle toho, ve kterém okamžiku průběhu procesu ovlivňují jeho tok.

- Událost typu *start* definuje místo, kde má proces nebo choreografie počátek. V případě sekvence toků tímto elementem začíná tok procesu,

z toho důvodu mu nemůže předcházet jiná událost. Událost typu *start*, podobně jako průběžná událost, může mít definovány specifické okolnosti (spouštěče), za kterých je proces zahájen. Symbol kružnice bez doplňujících značek uvnitř se používá k určení počátku procesu nebo choreografie, kde není příčina zahájení jasně definována.

- *Průběžné události* se mohou vyskytovat mezi začátkem a koncem procesu. Mají vliv na tok procesu nebo choreografie, ale nemohou proces spustit nebo ukončit. *Průběžné události* se používají pro sledování a kompenzaci chyb a výjimek. Dále pak definují místa v procesu, kde očekáváme příjem zpráv nebo kde dochází k odesílání zpráv. Mohou též pokrývat případy očekávaných zpoždění v rámci procesu. Události mohou být součástí běžného toku procesu, ale lze je také připojit k aktivitě, kde definují událost, při jejímž výskytu dojde k realizaci nebo přerušení aktivity.
- Událost typu *konec* definuje místo kde je proces nebo choreografie ukončena. Podobně jako ostatní typy událostí, mohu i zde nastat specifické okolnosti, analogické k událostem spouštějícím událost typu *start*, za kterých je proces ukončen.

Aktivity. Aktivita je obecný pojem pro práci, která je prováděna v rámci procesu. Aktivity mohou být atomické – *úlohy*, nebo složené – *podprocesy*. Aktivity reprezentují místa v procesu, kde se provádí práce.

- *Úloha* je atomická aktivita uvnitř procesu. *Úloha* se používá v případě, kdy není nutné nebo možné aktivu modelovat detailněji.
- *Úloha v choreografii* je opět atomická aktivita uvnitř choreografie. Reprezentuje množinu jedné nebo více výměn zpráv. Každá úloha se skládá ze dvou účastníků.
- *Podproces* je složená aktivita, která je součástí procesu nebo choreografie. Obvykle se v diagramu vyskytuje ve sbalené podobě, která neumožňuje zobrazit podrobnosti o dílčím procesu uvnitř. Sbalený podproces je možné rozbalit a zobrazit tak detailní pohled na dílčí proces. Veškeré sekvenční toky dílčího procesu musí být umístěny uvnitř symbolu pro podproces.
- *Vnořená choreografie (Sub-Choreography)* vychází ze stejných principů jako podproces, pouze se liší grafickým symbolem.

Brány. Brány se používají pro řízení, rozdělování a slučování sekvenčního toku v procesech nebo choreografiích. Všechny typy bran mohou tok jak rozdělovat, tak i slučovat, a jsou značeny shodně symbolem prázdného kosočtverce. Pomocí vnitřních symbolů se specifikuje jejich chování.

- *Vylučovací brány* umožňují větvit tok procesu na základě rozhodnutí založeného buď na datech, nebo na událostech. *Vylučovací brány* jsou křižovatkami uvnitř procesu, ze kterých mohou vést dvě i více cest, z nichž může být vybrána právě jedna, po které bude tok procesu pokračovat.
    - Nejčastěji používaným druhem brány je *vylučovací brána založená na datech*. Rozhodování probíhá na základě splnění podmínky, která je formulována jako "běžná otázka", která má předem taxativně určenou množinu možných odpovědí, reprezentující jednotlivé větve procesu.
    - *Vylučovací brány založené na událostech* představují body v procesu, kde dochází k jeho větvení na základě událostí, které nastanou právě v tomto bodě. Události, na jejichž základě se rozhodne (obvykle přijetí zprávy), jsou místěny za branou na jednotlivých větvích toku. První událost, která je aktivována, určuje větev, kterou bude proces dále pokračovat.
- *Paralelní brány* se používají k vytváření a synchronizaci paralelních toků. Symbol *paralelní brány* se liší od vylučovacích bran a na rozdíl od nich musí vždy symbol kosočtverce obsahovat doplňkovou značku "+", aby byl odlišitelný od ostatních druhů bran. Při použití tohoto typu brány dojde vždy k rozdělení toku na více cest, aniž by musely být splněny nějaké předem stanovené podmínky. Při použití brány pro synchronizaci více paralelních toků brána vždy čeká, dokud nepřijme všechny vstupní toky, a až poté spouští toky na svém výstupu.

4.2.2 Data

Pro modelování dat v rámci toku procesu se primárně používají datové objekty. Datové objekty poskytují informace o tom, co aktivity vyžadují pro své vykonávání a jaká data produkují. Životní cyklus datových objektů je vázán na životní cyklus procesu nebo subprocesu, ve kterém je obsažen.

4.2.3 Spojovací objekty

Existují čtyři základní způsoby jak propojovat tokové objekty navzájem nebo s některými ostatními prvky. Rozlišujeme čtyři druhy spojovacích objektů: *sekvenční toky*, *toky zpráv*, *asociace* a *datové asociace*.

Sekvenční toky. Určují pořadí, v jakém budou jednotlivé tokové objekty v rámci procesu nebo choreografie vykonávány. Každý sekvenční tok má právě jeden zdroj a právě jeden cíl a nemůže překračovat hranice podprocesu nebo bazénu.

Muže mít definovanou podmínku, při jejímž splnění bude proces pokračovat právě touto větví. Podmínky se většinou využívají v případech, kdy je zdrojem sekvenčního toku brána nebo aktivita.

Toky zpráv. Používají se k zobrazení toku zpráv mezi dvěma účastníky procesu, kteří jsou připraveni zprávy odesílat a přijímat. Každý účastník procesu je zobrazen jako samostatný bazén. Toky zpráv se mohou dotýkat hranic bazénů nebo jsou napojeny na tokové objekty uvnitř bazénu. Nemohou však spojovat dva tokové objekty v rámci jednoho bazénu.

Asociace. Používá se k propojení informací a artefaktů s ostatními elementy, nejčastěji textovými popisy.

Datové asociace. Datové asociace se využívají pro přesun dat mezi datovými objekty a vstupy a výstupy aktivit a procesů. Účelem získávání dat z datových objektů a procesních datových vstupů je naplnit vstupy aktivit a později po provedení aktivity vrátit výsledné výstupní hodnoty zpět do datových objektů nebo procesních datových výstupů.

4.2.4 Dráhy

Bazén. Účastníci procesů jsou v BPMN reprezentováni bazény. Na bazén takém můžeme pohlížet také jako na dráhu, která odděluje množinu činností od ostatních bazénů, což se obvykle vyskytuje v případě modelování B2B procesů. Bazén může obsahovat procesy, ale může také zůstat prázdný a vystupovat jako černá skříňka, u které známe pouze vstupy a výstupy. Účastník procesu může zastávat obecnější roli (výrobce, prodávající, kupující) nebo jím může být konkrétní subjekt či osoba (společnost, účetní, ředitel). Interakce mezi bazény zajišťují tokové zprávy, které nelze použít pro interakce uvnitř bazénu. Vnitřní interakce jsou zajišťovány pomocí sekvenčních toků, které ale nelze použít pro vnější interakce.

Dráha. Dráhy rozdělují bazén na menší části, z nichž každá může představovat nějakou dílčí roli v rámci organizace. Uvažujeme-li bazén jako jeden proces, lze pomocí drah kategorizovat a organizovat jednotlivé aktivity. Na rozdíl od bazénu mohou sekvenční toky přecházet přes hranice drah.

4.2.5 Artefakty

Artefakty se používají pro doplnění informací o procesu, které buď nebylo možné zahrnout do procesu, nebo které slouží k lepšímu pochopení modelované problematiky. BPMN definuje dva standardizované artefakty – *Group* neboli *Skupina* a *Text annotation*, tedy textový popis. Návrhářům procesů a tvůrcům

modelovacích nástrojů je dovoleno definovat další artefakty podle potřeby. Nesmějí však být v konfliktu s prvky, které jsou již v notaci definovány.

Skupina. Skupinu tvoří množina grafických prvků, které jsou ve stejné kategorii. Tento typ seskupování prvků nemá vliv na sekvenční toky uvnitř skupiny. Kategorie, jejichž sémantika je definované uživatelem, lze využít pro dokumentaci nebo k analytickým účelům.

Textový popis. Textové popisy slouží návrhářům procesů pro doplnění diagramů o informace užitečné a podstatné pro jejich uživatele.

4.3 Business Object Relation Modeling

První práce popisující možnost sloučení objektově orientovaného přístupu a popisu objektů a jejich chování pomocí konečných automatů, byla kniha autorů Shlaerové a Mellora (Shlaer a Mellor, 1992). Výhody, možnosti a postupy použití objektově orientovaného přístupu pro modelování obchodních procesů popisuje Taylor (Taylor, 1995), který podrobně rozebírá základní principy objektového přístupu v kontextu obchodních procesů. Možnost využít objektově orientovaného přístupu pro popis obchodních procesů, někdy též označovaných jako podnikové procesy, jak ji popisuje Taylor (Taylor, 1995) spolu s prací Shlaerové a Mellora (Shlaer a Mellor, 1992), poskytly teoretický a konceptuální rámec pro vznik metody BORM (Knott, a další, 2006).

Metoda BORM byla vyvíjena postupně od roku 1993. Původním záměrem bylo vytvořit metodiku orientovanou na podporu tvorby objektově orientovaných softwarových systémů založených na čistých objektově orientovaných programovacích jazycích a nerelačních objektových databázích. V průběhu vývoje bylo zjištěno, že je možné využít některé definované techniky pro modelování procesů obecně. BORM pokrývá všechny fáze vývoje softwaru a je ho možné využít nejen při tvorbě softwaru, ale i k analýze požadavků na projektovaný systém a na modelování obchodních procesů. Od roku 1996 je další vývoj podporován firmou Deloitte&Touche Czech Republic and Central Europe, kde je tato metoda také používána. Pro metodu BORM jsou podle (Merunka, 2008) charakteristické následující vlastnosti:

- BORM je navržen jako metoda, která pokrývá všechny fáze vývoje softwaru. Velká pozornost je v BORMu věnována úvodním fázím projektu a postupům, jak najít objekty v zadaném problému a zkontrolovat jejich správnost. Techniky z těchto fází BORMu lze používat samostatně pro modelování procesů i takových systémů, které nemají přímý vztah k tvorbě softwaru. Jinými slovy lze BORM použít pro mapování procesů problémové

domény, jejich případnou optimalizaci a tím rozpoznat a případně zvýšit CMMI (Capability Maturity Model Integration) procesu, aniž by výsledek vyústil v konkrétní softwarovou implementaci, která bude procesy podporovat.

- BORM pro každou jednotlivou fázi životního cyklu využívá v diagramech jen omezenou sadu pojmů. Předpokládá se totiž, že během projektování dochází k postupným přeměnám pojmů na jiné. Například ve fázi analýzy se nepoužívají pojmy jako agregace, jednoduchá či vícenásobná dědičnost, protože tyto pojmy jsou relevantní až pro implementaci. Naopak pojmy jako stav, přechod či asociace jsou používány během analýzy, ale ve fázi implementace, kdy se snažíme model přizpůsobit cílovému implementačnímu prostředí, se s nimi již nepracuje. Nejde jen o postupné zvyšování úrovně detailu ve vytvářeném modelu, ale skutečně o řadu transformací modelu v průběhu životního cyklu.
- V BORMu je každý pojem reprezentován shodnými symboly bez ohledu na to, jestli se jedná např. o diagramy datové struktury nebo komunikaci mezi objekty. 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 BORMu 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 (Merunka, 2008).

Metoda BORM má podle (Knott, a další, 2006) následující výhody:

- 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 zdrojem požadavků na systém a dále poskytuje nástroj pro analýzu a návrh softwaru.
- 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.
- Koncoví uživatelé, nebo obecně účastněné strany problémové domény, jsou schopni velmi rychle pochopit přístup a diagramy používané v BORM.
- Podle zkušeností z Deloitte & Touche je analytická fáze projektu v BORM, v porovnání s ARIS, přibližně třikrát až čtyřikrát rychlejší.
- 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.
- 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.

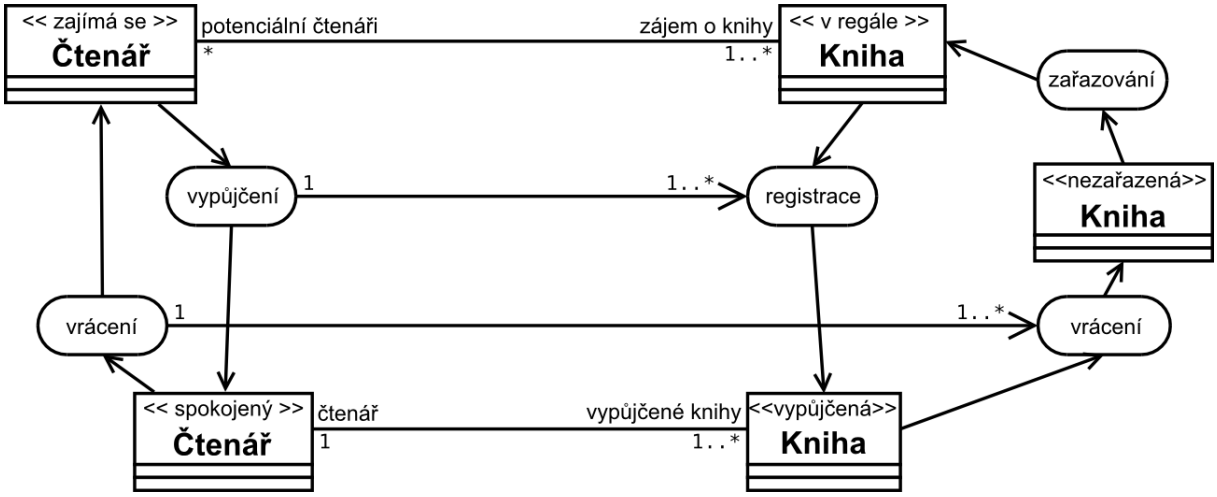
4.3.1 Rozšíření BORM

Na práci Shlaerové a Mellora (Shlaer a Mellor, 1992), která byla zmiňována v souvislosti s původní koncepcí metody BORM, navazuje Merunka (Merunka, 2012), 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 a Mellor, 1992) a (Taylor, 1995) se shodují, že při tvorbě procesního modelu je důležité nejprve identifikovat stavy, ve kterých se jednotliví participanti[1] 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 (Merunka, 2012).



Obr. 4.2. UML podoba procesních diagramů BORM – vlastní zpracování autora podle (Merunka, 2012)

Možné výhody nového přístupu s využitím prvků UML, jak je uvádí (Merunka, 2012):

- Oproti diagramům v BORM, reprezentovaných pomocí výše zmiňované podoby jazyka UML, pokrývají BPMN a UML pouze část celého

[1] Participanty jsou v celé práci myšleni jak externí tak interní účastníci procesů. Definice praticipanta z pohledu BORM je uvedena v kapitole 10.1.

využitelného prostoru, který umožňuje objektově orientovaný přístup modelování.

- Nejdůležitějšími pojmy jsou stavy objektů. Aktivity představují "pouze" způsob, jak přejít z jednoho stavu do stavu jiného. Při tvorbě procesních modelů nebo návrhu softwarových komponent by se mělo začínat identifikací stavů, v nichž se jednotliví participanti v průběhu času mohou nacházet. Modelování procesů pak může být 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.

4.3.2 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 prvotního 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 vnikla 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 (Merunka, 2012).

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 a Mellor, 1992). Jak uvádí (Goldber a 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 "hmotné" 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 "hmotných" 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 (Rubin a Goldberg, 1992). Použití tohoto základního principu je podmíněno úplnou a formálně správnou specifikací problémové domény, která zvláště pro velké systémy není vždy k dispozici. Řešení nabízejí Rubin a Goldbergová (Rubin a Goldberg, 1992) 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 (Goldber a 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.

Autoři OBA (Rubin a Goldberg, 1992) uvádějí následujících pět kroků, ze kterých se metoda skládá:

- **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.
- **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.
- **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.
- **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:

- **Abstrakce** – Mají-li dva nebo více objektů definované stejné služby nebo logické vlastnosti, vytvoříme nový objekt, kterému tyto vlastnosti nebo služby přiřadíme. Zároveň tyto vlastnosti nebo služby od původních objektů odebereme.
- **Specializace** – 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 od původních objektů odejmeme a vytvoříme objekt nový, kterému tyto služby nebo logické vlastnosti přiřadíme. Na rozdíl od abstrakce, v případě specializace přiřadíme původním objektům atribut obsahující nově vytvořený objekt, kterému byly odejmuté vlastnosti nebo služby přiřazeny.
- **Faktorizace** – 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 pak přiřadíme podmnožinu atributů nebo služeb původního objektu. Původní objekt pak můžeme vyřadit z množiny objektů. Cílem je vytvořit z jednoho "univerzálního" objektu více specializovaných objektů.

- **Modelování dynamiky systému.** Předchozí kroky popisovaly strukturu systému staticky, tedy v jednom místě a čase. Poslední krok popisuje 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 (Rubin a Goldberg, 1992), modifikována. Jednotlivé kroky použití OBA pro analýzu procesního modelu podle BORMu, jak je uvádí (Merunka, 2008), jsou následující:

- **Rozpoznání procesů.** V tomto kroku 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 jako detailního popisu 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.
- **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.
- **Modelování procesů.** Zde 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.
- **Verifikace a validace.** Zde se kontroluje shoda mezi diagramy, tabulkami a skutečnými požadavky na systém. K tomu slouží dva nástroje. Jedním z nich je datový model obsahující skutečná data, pomocí nichž 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.

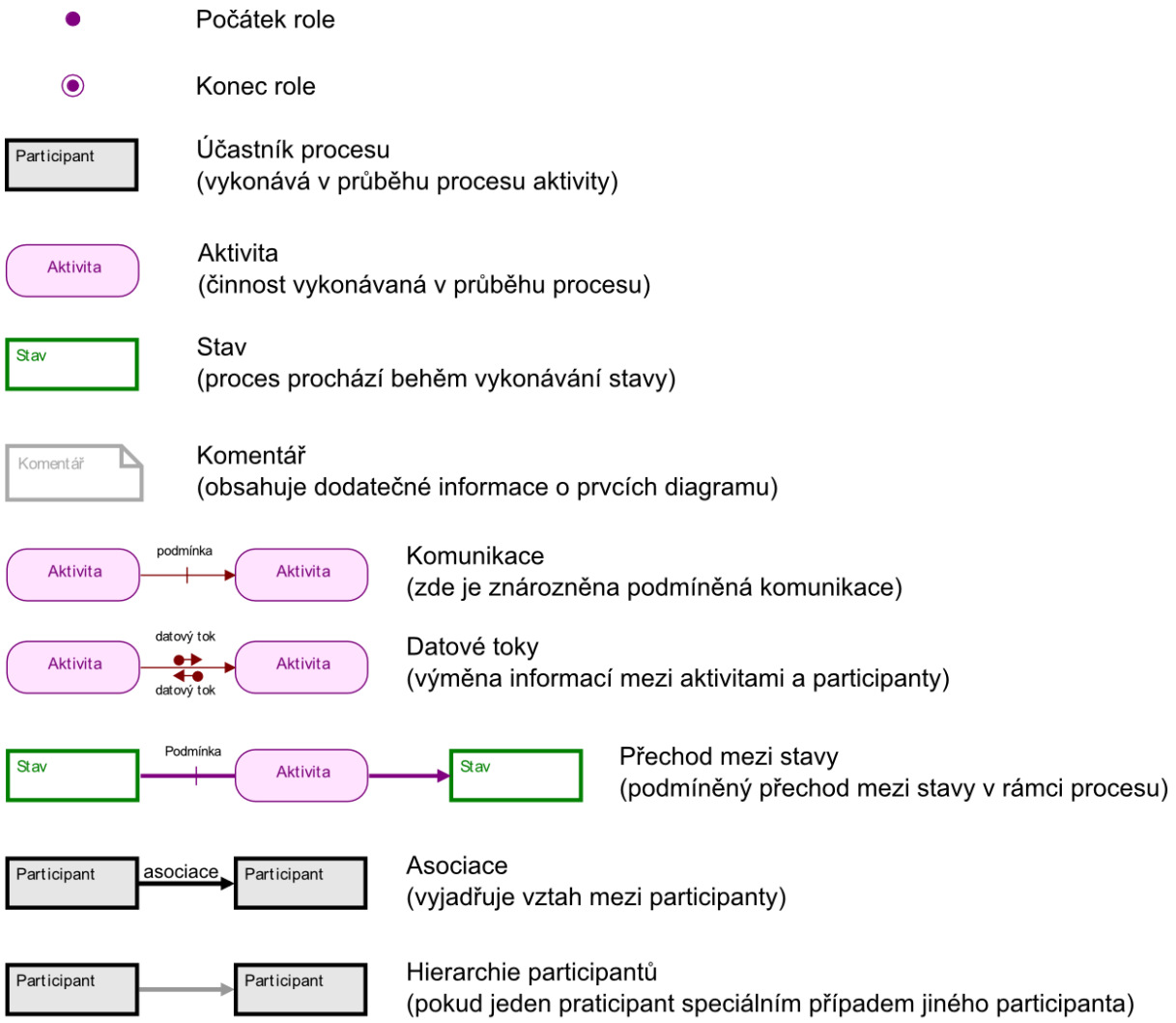
4.3.3 Procesní diagramy v BORM

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, jež jsou plně postačující pro prvotní popis modelovaných procesů, a tím je použitelný i pro konzultace se zadavateli/zákazníky (Knott, a další, 2000).

Jak uvádí (Merunka, 2000), ORD dovoluje modelovat jednotlivé procesy současně dvojím způsobem:

- Sekvence stavů a přechodů každého objektu, na které lze nazírat jako na jednotlivé stavové diagramy, vyjadřující 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. Nazíráme-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 CraftCASE 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).



Obr. 4.3. Základní grafické elementy notace BORM – vlastní zpracování autora

Jak ukazuje obrázek 4.3, základními prvky procesního diagramu jsou:

- *Participant* – reprezentuje účastníka procesu a je zobrazen jako černý orámovaný obdélník
- *Stav* – každý participant může v čase měnit svůj stav na základě vykonaných činností. Stavy jsou reprezentovány zeleným obdélníkem
- *Činnost nebo aktivita* – představuje činnosti vykonávané participantem. Slouží pro přechod z jednoho stavu do jiného. Pro aktivity je použit fialový 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.

5 Konečné automaty

Konečné automaty lze použít pro modelování různých problémů, mezi něž patří návrh komunikačních protokolů, modelování logických obvodů, lexikální analyzátory, umělé inteligenci. Uplatnění též nalézají v biologii či popisu přirozených jazyků (Gill, 1962).

Existuje mnoho způsobů modelování chování systémů. Jako jeden z nejstarších a nejznámějších uvádějí (Gill, 1962) a (Wright, 2005) stavové automaty. Základní kameny teorie automatů položili (Huffman, 1954), (Mealy, 1955) a (Moore, 1956), z jejichž prací vycházejí všichni pozdější autoři. Konečné automaty dovolují zachytit a popsat stav systému nebo procesu v určitém místě respektive čase. Chování celého systému je možné odvodit na základě jeho stavů. Použití této techniky není omezeno volbou konkrétního modelovacího nástroje a je nezávislé na implementaci.

Obdobnou techniku můžeme použít pro modelování a návrh softwarových systémů respektive vytváření procesních modelů. Identifikací stavů, ve kterých se systém může nacházet, jaké vstupy nebo události vyvolávají přechody z jednoho stavu do jiného a jak se systém bude chovat v každém svém stavu, je možné popsat chování jednotlivých participantů procesu.

Aby bylo možné použít pro popis systému nebo procesu konečné automaty, musí mít tento podle (Wright, 2005) následující vlastnosti:

- Musí být popsatelný pomocí konečné množiny stavů
- Musí mít omezenou sadu vstupů nebo událostí, které mohou vyvolat přechod mezi stavy
- Chování systému v daném okamžiku závisí na konkrétním stavu a vstupu nebo události, které v daném okamžiku nastanou.
- Pro každý stav, ve kterém se systém může nacházet, je chování definováno množinou možných vstupů
- Systém má definován počáteční stav

Konečný automat definuje Vaníček (Vaníček, 2007) jako abstraktní model výpočtu realizovaný počítačem. Je koncipován jako abstraktní stroj, přičemž ve kterémkoli okamžiku se může nacházet pouze v jednom konkrétním stavu, který je dán konečnou množinou všech možných stavů. Změna aktuálního stavu na stav jiný je možná pomocí podmínek nebo událostí, které jsou souhrnně označovány jako přechody. Obecně je konečný automat zejména definován pomocí

množiny stavů a podmínek, za kterých jsou spouštěny jednotlivé přechody. Kromě návrhu počítačových programu nebo algoritmů se využívají konečné automaty pro návrh sekvenčních logických obvodů.

Formálně je konečný automat M uspořádaná pětice (K, Σ, δ, q_0, F), kde:

- K je konečná neprázdná množina všech stavů automatu.
- Σ je konečná neprázdná vstupní abeceda automatu.
- δ: $K \times \Sigma \rightarrow K$ je přechodová funkce, která každému (současnému) stavu automatu a každému (právě zpracovávanému) symbolu vstupní abecedy přiřazuje nový stav automatu.
- $q_0 \in K$ je počáteční stav automatu. V tomto stavu „práce“ automatu začíná.
- $F \subseteq K$ je neprázdná konečná množina koncových stavů. Pokud automat ukončí svoji práci ve stavu, který patří do F, odpovídá kladně na otázku položenou na vstupu (slovo přijímá).

Dále lze říci:

- Slovo nad abecedou Σ je sekvence symbolů vstupní abecedy Σ konečné délky.
- Analogicky pak jazyk nad abecedou Σ je množina slov nad abecedou Σ.
- Přechodovou funkci $\delta(q,x) = p$ lze interpretovat: Konečný automat M ve stavu q a při vstupu x se po jeho zpracování dostane do stavu p.
- Slovo je automatem M rozpoznáno ze předpokladu, že $\delta(q,x) = p$ pro nějaké $p \in M$.
- Jazyk rozpoznatelný automatem M, označujeme $L(M)$, je konečná množina všech slov x rozpoznatelných automatem. Takový jazyk pak můžeme zapsat jako $L(M) = \{x \mid \delta(q_0, x) \in F\}$.

Konečný automat je považován za nástroj pro řešení úloh, přičemž úloha je postavena jako otázka, zda nějaké slovo reprezentované řetězcem znaků vstupní abecedy patří nebo nepatří do zvoleného jazyka. Konečný automat vychází vždy z počátečního stavu a postupně zpracovává znaky slova ze vstupu a na základě přechodové funkce δ mění svůj stav. Jakmile zpracuje celý řetězec znaků vstupního slova, dá odpověď ne, nebo ano, pokud dané slovo patří do množiny slov daného jazyka. Jinými slovy, pokud se automat po zpracování celého vstupního slova nachází v některém z koncových stavů, slovo do jazyka patří. Naopak pokud automat skončí v jiném, než koncovém stavu, slovo nebylo

rozpoznáno, a tedy do jazyka nepatří. Jazyk se nazývá rozpoznatelný konečným automatem, pokud existuje konečný automat, který rozpoznává tento jazyk, to znamená pro každé slovo je schopen rozhodnout, zda do jazyka patří, či nikoliv (Vaníček, 2007).

Pro potřeby modelování systému a procesů je konstrukce úlohy, jak byla popsána výše, nedostačující. Nevystačíme s prostým automatizovaným nalezením odpovědi ano či ne, ale potřebujeme zachytit transformaci vstupních informací na výstupy, tedy automatizované zpracování dat. Lze toho dosáhnout změnou modelu konečného automatu, kdy výsledkem činnosti automatu nebude pouze jeho stav, na jehož základě rozhodujeme, zda daný jazyk byl či nebyl rozpoznán, ale výstup ve výstupní abecedě generovaný automatem. Základními modely konečných automatů jsou Mooreův a Meallyho sekvenční stroje.

5.1 Mooreův sekvenční stroj

Mooreův stroj (Moore, 1956) lze definovat jako uspořádanou šestici $(Q, p, \Sigma, \Delta, \delta, \mu)$, kde:

- Q je konečná množina stavů,
- $p \in Q$ je počáteční stav,
- Σ je konečná neprázdná vstupní abeceda,
- Δ je konečná neprázdná výstupní abeceda,
- δ: $Q \times \Sigma \rightarrow Q$ je přechodová funkce (dvojici stávající stav, vstupní symbol přiřazuje nový stav),
- μ: $Q \rightarrow \Delta$ je značkovací funkce, která každému stavu přiřazuje výstupní symbol.

Mooreův stroj generuje výstup na základě značkovací funkce, tedy pouze na základě aktuálního stavu, ve kterém se nachází.

5.2 Meallyho sekvenční stroj

Meallyho sekvenční stroj (Mealy, 1955) lze definovat jako uspořádanou šestici $(Q, p, \Sigma, \Delta, \delta, \mu)$, kde:

- Q je konečná množina stavů,
- $p \in Q$ je počáteční stav,
- Σ je konečná neprázdná vstupní abeceda,
- Δ je konečná neprázdná výstupní abeceda,
- δ: $Q \times \Sigma \rightarrow Q$ je přechodová funkce (dvojici stávající stav, vstupní symbol přiřazuje nový stav),
- μ: $Q \times \Sigma \rightarrow \Delta$ je výstupní funkce (každému stavu a každému vstupnímu symbolu přiřazuje výstupní symbol).

Meallyho sekvenční stroj generuje výstup na základě vstupu, přechodové funkce a aktuálního stavu, ve kterém se nachází.

Základní rozdíl mezi oběma modely sekvenčních strojů spočívá v tom, že výstup Mooreova stroje závisí pouze na stavu, ve kterém se stroj právě nachází. Za předpokladu, že by byla vstupní abeceda pro oba modely stejná, Meallyho stroj může poskytovat v daném stavu různé výstupy v závislosti na aktuálně zpracovávaném symbolu, tedy v závislosti na vstupu. Z pohledu procesů jsou v případě Mooreova sekvenčního stroje vykonávané aktivity v rámci stavu (Mandrioli a Ghezzi, 1987), zatímco v případě Mealyho sekvenčního stroje jsou aktivity vázané na přechod mezi stavy (Davis, a další, 1994). Jak uvádí Vaníček (Vaníček, 2007), tento rozdíl není principiální. Každý Mooreův stroj je zřejmě Meallyho strojem. K Meallyho stroji lze sestrojit ekvivalentní Mooreův, který bude vykonávat stejnou funkci. Pokud každý stav Meallyho stroje rozštěpíme do tolika stavů, kolik je prvků vstupní abecedy, můžeme k němu sestrojit ekvivalentní Mooreův stroj.

Volba modelu závisí na aplikaci, způsobu, jakým bude automat realizován (při realizaci hardwarovými prostředky je obvykle použit model Mooreho automatu) a případně na osobních preferencí analytika případně programátora. Činnost konečného automatu lze realizovat oběma výše zmíněnými modely sekvenčních strojů. V praxi se často používají smíšené modely. Stavové automaty jsou již dlouho populární a představují efektivní způsob modelování dynamiky chování softwarových systémů (Gill, 1962) a (Wright, 2005).

5.3 Komunikující konečné automaty

Komunikující konečný automat (CFSM) je abstraktní model používaný pro potřeby specifikace návrhu, testování, validaci a syntézu komunikačních protokolů (Brand a Zafiropulo, 1983). Model popisuje procesy jako konečné automaty, které mezi sebou komunikují prostřednictvím teoreticky neomezeně velkého zásobníku tak, že prostřednictvím tohoto zásobníku asynchronně přijímají, respektive zasílají zprávy. Zásobník je tedy prostředkem komunikace a slouží k ukládání zpráv, které byly odeslány, ale nebyly dosud přijaty.

Hlavním problémem je zajistit, aby množina vzájemně komunikujících konečných automatů nebyla zatížena chybami v komunikaci natolik, že by nedocházelo k plynulému přecházení ze stavu do následujícího stavu u jednotlivých komunikujících automatů. Brand a Zafiropulo (Brand a Zafiropulo, 1983) uvádějí, že v zásadě existují tři oblasti nebo příčiny, kvůli kterým může dojít k přerušení komunikace. Jsou to nespecifikované přijímání zpráv, uváznutí procesu a nevymezené komunikace.

Peng a Puroshothaman (Peng a Puroshothaman, 1991) navazují na předchozí práci Branda a Zafiropula a představují přístup, jakým lze prostřednictvím analýzy datových toků jednotlivých automatů nastavit takové podmínky pro komunikace, které zabraňují nespecifikovanému přijímání zpráv a také uváznutí procesu. Každý proces by měl být zastoupen pouze jedním konečným automatem a zásobníky, tedy komunikační kanály by měly být používány pouze pro realizaci komunikace. Každá dvojice komunikujících procesů je spojena plně duplexním kanálem typu FIFO. S těmito omezeními je, jak uvádí (Brand a Zafiropulo, 1983), model hůře použitelný pro modelování procesů než například Petriho sítě. Na tuto práci navazuje Peng a Puroshothaman (Peng a Puroshothaman, 1991), kteří toto omezení odstraňují a zaměřují se na modelování procesů a ne na aplikaci v oblasti návrhu komunikačních protokolů.

Model komunikujících konečných automatů lze použít i v jiných oblastech, než jen pro potřeby syntézy komunikačních protokolů. Zmiňovaný článek (Peng a Puroshothaman, 1991) prezentuje příklad dvou vzájemně komunikujících procesů, modelovaných pomocí CFSM, u nichž je možné na základě analýzy jejich komunikační části bezpečně stanovit podmínky komunikace a přiřazení stavů. Výsledná implementace pak odráží strukturu řídicích a komunikačních toků získané z množiny komunikujících konečných automatů. Na základě (Peng a Puroshothaman, 1991) tedy lze využít CFSM pro popis průběhu nejen dvou procesů, ale využijeme-li tzv. síť komunikujících konečných automatů, také celé skupiny procesů. Nutno podotknout, že model procesů (ať už je

reprezentován skupinou konečných automatů nebo libovolným „procesním“ diagramem) zachycuje pouze události a stavy, které analytik nebo návrhář procesů předpokládá, že mohou nastat. Množina možných událostí a stavů, kterou uvažujeme je tedy vždy konečná.

Intuitivně je CFSM orientovaný graf s definovaným počátečním stavem (uzlem), kde každá jeho hrana odpovídá právě jedné aktivitě (události). Aktivity přijímají a odesílají zprávy z konečné množiny všech zpráv. Koncept uveřejněný v (Peng a Puroshothaman, 1991) předpokládá, že komunikace mezi dvěma a více CMFS je asynchronní. Důsledkem toho je nutnost využití již zmiňovaného, teoreticky nekonečně velkého zásobníku typu FIFO pro ukládání zpráv čekajících na zpracování. Mezi každou dvojicí vzájemně komunikujících CFSM musí být tedy vložen tento zásobník. Naproti tomu (Gouda a Yu, 1984) hovoří o speciální třídě komunikujících konečných automatů, u kterých dochází k odeslání zprávy jedním procesem a jejím přijetím procesem druhým v nulovém čase. Ve stejném okamžiku, kdy je zpráva jednou ze stran odeslána, je druhou stranou přijata a zpracována. Automat postrádá prvek asynchronní komunikace. Nejsou proto zapotřebí žádné kanály pro ukládání zasílaných zpráv a problém návrhu automatů se tím velmi zjednodušuje. Zasílání zpráv není v případě této zvláštní třídy CFSM nijak omezeno a stejně tak nedochází k nespecifikovanému přijímání zpráv. Může docházet k uváznutí procesu a automat také může obsahovat nespustitelné přechody. Nutno uvést, že zmiňovaná práce (Gouda a Yu, 1984) byla publikována o sedm let dříve než (Peng a Puroshothaman, 1991) a primárně se zabývá syntézou dvou komunikujících konečných automatů, které si vyměňují zprávy prostřednictvím dvou jednocestných FIFO kanálů.

Formálně (Peng a Puroshothaman, 1991) definují komunikující konečný automat P_i jako uspořádanou čtveřici $\left(S_i, \left\langle \Sigma_{i,j} \right\rangle_{j \in I} \cup \left\langle \Sigma_{j,i} \right\rangle_{j \in I}, \delta_i, p_{0i}\right)$, kde:

- S_i je konečná neprázdná množina stavů automatu i
- I je množina všech komunikujících konečných automatů tvořící síť, $I = \{1, \ldots, n\}$, kde $n \geq 2$.
- $\Sigma_{i,j}$ je množina druhů zpráv, které může automat P_i odeslat automatu P_j a $\Sigma_{j,i}$ je množina druhů zpráv, které může automat P_i přijmout od automatu P_j. Předpokládáme, že $\Sigma_{i,i} = \emptyset$ dokud P_i neodešle respektive nepřijme zprávu, kterou si sám odeslal.

- Nechť $-\Sigma_{i,j} = \{-m \mid m \in \Sigma_{i,j}\}$ a $+\Sigma_{j,i} = \{+m \mid m \in \Sigma_{j,i}\}$. δ_i je částečné zobrazení, $\delta_i : S_i \times \left(\left\langle \Sigma_{i,j} \right\rangle_{j \in I} \cup \left\langle +\Sigma_{j,i} \right\rangle_{j \in I}\right) \times I \to 2^{S_i}$. 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.
- p_{0i} je počáteční stav automatu P_i.

Pro potřeby této práce je výše uvedená definice postačující a nebudeme se zde zabývat zápisem asynchronních komunikací přes zásobníky typu FIFO. Asynchronní komunikace nevyžadují, na rozdíl od autorem této práce uvažovaného konceptu, přijetí zprávy ve stejné chvíli kdy je odesílána, a pokud bude velikost kanálu provázána s konkrétním stavem, lze stále výsledný model simulovat pomocí konečného automatu (Peng a Puroshothaman, 1991).

Ačkoli nejstarší z výše uvedených prací (Brand a Zafiropulo, 1983) byla publikována již před třiceti lety, patří stále mezi hlavní zdroje informací z oblasti teorie komunikujících konečných automatů, na niž další autoři (Gouda a Yu, 1984) a (Peng a Puroshothaman, 1991), zabývající se obdobnou problematikou, navazují a odkazují.

6 Webové služby

Definice webové služby podle (W3C WS-Architecture, 2004):

> *"Webová služba je softwarový systém navržený pro podporu vzájemné interakce dvou systémů v síti. Rozhraní má popsané ve strojově čitelné podobě (konkrétně WSDL). Ostatní systémy interagují s webovou službou předepsaným způsobem za použití SOAP zpráv, obvykle předávaných pomocí HTTP protokolu"*[1]

Definice webové služby podle (Cerami, 2002):

> *„Webová služba je jakákoli služba, která je k dispozici na internetu, používá standardizovaný systém založený na XML pro zasílání zpráv a není vázána na konkrétní operační systém nebo programovací jazyk."*[2]

Oba výše zmiňovaní autoři (W3C WS-Architecture, 2004) a (Cerami, 2002) shodně uvádějí, že webová služba je abstraktní prvek, který musí být implementován konkrétním agentem. Agentem může být software nebo hardware, který přijímá a odesílá zprávy. Služba je charakterizována abstraktní množinou funkcí, které poskytuje svému okolí. Jedna webová služba může být implementována dvěma různými agenty, přičemž každého z agentů lze implementovat v jiném programovacím jazyce. Přestože se mnohou agenti změnit, webová služba zůstává stejná. Cílem webové služby je poskytovat nějaké funkce jménem svého majitele. Majitelem může být osoba nebo organizace. Aby mohlo k výměně zpráv docházet, je nutné, aby se žadatel a poskytovatel služby shodli na sémantice a mechanizmu výměny zpráv.

Ačkoli to není pro fungování webových služeb nezbytné, (Cerami, 2002) uvádí následující užitečné vlastnosti, které je vhodné implementovat:

- Webová služba by měla být sebepopisná. Vytvoříme-li novou, veřejně dostupnou službu, měli bychom zároveň zveřejnit popis rozhraní této služby. Dále bychom měli ke službě připojit dokumentaci, která umožní

[1] A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. (W3C WS-Architecture, 2004)

[2] A web service is any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any one operating system or programming language. (Cerami, 2002)

vývojářům službu implementovat. Pokud jsme vytvořili službu na bázi SOAP, měla by zahrnovat veřejné rozhraní zapsané v běžné XML gramatice. Gramatika by měla popisovat veřejně dostupné metody služby, argumenty a návratové hodnoty metod.

- Webová služba by měla být zjistitelná. Vytvoříme-li webovou službu, měl by existovat relativně jednoduchý mechanizmus pro její zveřejnění. Stejně tak by měl existovat jednoduchý mechanismus, s jehož pomocí by zájemci mohli službu a její veřejné rozhraní najít. Přesný mechanizmus by mohl být realizován prostřednictvím zcela decentralizovaného systému nebo logičtěji centralizovaného systému.

V architektuře webových služeb podle (W3C WS-Architecture, 2004) a (Cerami, 2002) existují tři hlavní role:

- *Poskytovatel služby* – realizuje služby a dává je k dispozici prostřednictvím internetu
- *Žadatel* – osoba nebo organizace, která využívá existující webové služby. Zasílá požadavky v XML formátu a přijímá odpovědi.
- *Registr služeb* – logicky centralizovaný adresář služeb, kde mohou jednotliví vývojáři publikovat nové veřejné služby.

6.1 Web Service Modelling Framework

Webové služby posunuly internet, nebo lidově řečeno web, od prostoru pro sběr informací k distribuovanému výpočetnímu nástroji. Aby bylo možné využívat potenciál webových služeb naplno, bylo nutné dále rozvíjet prostředky pro jejich popis. Jednou z prvních iniciativ pro vytvoření plnohodnotného rámce pro modelování webových služeb bylo definování WSMF (Web Service Modeling Framework). Výchozím zdrojem pro popis WSMF je (Fensel a Bussler, 2002).

Mnoho jazyků pro popis webových služeb rozlišuje mezi základními a komplexními webovými službami. Základní webové služby jsou jednoduché vstupně výstupní prvky, zatímco komplexní webové služby pokrývají celý proces až do dílčích úkolů a často spolupracují s dalšími webovými službami. Je nutné rozlišovat mezi vnitřní složitostí a složitostí rozhraní, nebo lépe řečeno složitostí pozorovatelného chování. Jak uvádí (Fensel a Bussler, 2002), rozdíl mezi vnitřní strukturou a rozhraním může vypadat jako triviální, ale má dva závažné důsledky:

- Některé přístupy pro popis webových služeb výslovně nerozlišují mezi vnitřním popisem služby a popisem jejího rozhraní. Poskytují prostředky

pro popis jako diagramy datových a řídicích toků, aniž by jasně upřesnily, zda tyto diagramy chápat jako popis rozhraní pro přístup ke službě nebo jako vnitřní popis webové služby. Vnitřní složitost služby je často odrazem Business intelligence poskytovatele, což je hlavní rozdíl mezi vnitřním popisem workflow služby a popisem jejího rozhraní.

- Dichotomie mezi základními a komplexními webovými službami je příliš zjednodušující. Na různé stupně složitosti popisu webových služeb nelze nahlížet pouze ze dvou extrémů. Často se začíná popisem základní funkčnosti služby, která je postupně rozšiřována o další prostředky popisující různé aspekty webových služeb.

WSMF popisuje webové služby jako černé skříňky, tedy nezabývá se jejich vnitřní strukturou. Každá webová služba musí mít název, tedy unikátní identifikátor. Její popis by měl obsahovat informace o tom, jaká data jsou očekávána na vstupu a jaká data může koncový uživatel od služby získat.

Informace, které jsou od koncového uživatele vyžadovány pro spuštění služby, označuje WSMF jako vstupní podmínky (pre-condition). Například máme-li službu, která poskytuje aktuální informaci o venkovní teplotě, je nutné, aby služba nejprve dostala informaci o městě nebo regionu, pro který chceme zjistit aktuální teplotu. Bez upřesnění informace o poloze by nemělo smysl žádat službu o odpověď.

Očekávané výstupy (post-conditions) popisují, jaký bude výstup v závislosti na vstupních informacích. Použijeme-li stejný příklad jako v případě vstupních podmínek, získáme odpověď na požadavek venkovní teploty v nějaké lokalitě v podobě konkrétní teploty.

Popis webové služby dále obsahuje informace o struktuře vstupních a výstupních dat. Vstupními porty jsou data přijímána a postoupena dále do komplexní služby. Vstupní porty se chovají jako proměnné, jejichž obsah je službě předáván v průběhu jejího zpracování. Data mohou být službou přijímána také v průběhu jejich zpracování. Analogicky lze popsat výstupní porty, kterými webové služby poskytují odpovědi na požadavky. Podobně jako v případě vstupních portů, mohou výstupní porty poskytovat informace průběžně před úplným dokončením činnosti služby. Výhodou je, že koncový uživatel nemusí čekat, až bude komplexní služba dokončena a data jsou mu k dispozici, jakmile je to možné.

Chybová hlášení mohou komplexní webové služby poskytovat prostřednictvím chybových portů, kdykoli se vyskytne chyba. Lze definovat více chybových portů,

pomocí nichž lze informovat o chybách, které se vyskytnou různých okamžicích během vykonávání služby.

Jak shodně uvádějí (Fensel a Bussler, 2002) a (Roman, a další, 2005), WSMF se skládá ze čtyř odlišných hlavních prvků:

- *Ontologie* – Ontologie jsou klíčovou technologií sémantického webu. Definují formální sémantiky informací, tedy umožňují informace počítačově zpracovat. Dále umožňují přiřadit počítačově zpracovatelnému obsahu význam pochopitelný pro člověka na základě dohodnuté terminologie.
- *Cíle (Goals)* – Cíle definují stav, kterého by mělo být dosaženo prostřednictvím webových služeb. Specifikace cíle se skládá ze vstupních podmínek (pre-conditions) a očekávaných výstupů (post-conditions), o kterých již bylo hovořeno. Cíle by měly být odděleny od popisu webové služby, protože stejná webová služba naplňuje různé cíle a naopak různé služby mohou naplňovat stejný cíl.
- *Popis webových služeb* – Definuje různé aspekty webových služeb jako popis rozhraní, mechanizmy kompenzace chybějících dat, ošetření výjimek a další.
- *Adaptéry (Mediators)* – Adaptéry mají velký význam pro komponentovou tvorbu softwaru. Pro otevřené a flexibilní prostředí webově orientovaných výpočtů jsou adaptéry základním prostředkem, jak se vyrovnat s heterogenitou prostředí.
    - *Adaptéry datových struktur* – Webová služba může poskytovat vstup pro jinou službu, ne však ve formátu, který druhá služba očekává.
    - *Adaptéry obchodní logiky* – Dvě webové služby poskytují komplementární funkcionalitu. Mohou být v zásadě propojeny, pokud bude jejich obchodní logika kompatibilní.
    - *Adaptéry protokolů pro výměnu zpráv* – Webové služby se mohou lišit ve způsobu, jakým zajišťují spolehlivou komunikaci. Například SOAP přes HTTP je nespolehlivý a vyžaduje od obou komunikujících stran, aby na rámec protokolu implementovaly potvrzování přijetí zpráv, překročení časového limitu pro odpověď, opakované odesílání zpráv a další.

- *Adaptéry pro dynamické volání služeb* – Webová služba může volat další služby a tím integrovat jejich funkčnost. Lze toho dosáhnout dvěma způsoby. Je možné dopředu fixně definovat, kdy bude jaká služba volána, což není příliš flexibilní přístup. Druhou možností je odkazovat na určité cíle, které externí služby zprostředkovávají a během vykonávání služby dynamicky podle potřeby volat další služby.

V mnoha případech je nezbytné popsat rozhraní webových služeb komplexněji. Na základě (Fensel a Bussler, 2002) lze rozlišit následující oblasti, kvůli kterým může být vyžadován komplexnější popis rozhraní:

- *Zpracování výjimek a chyb* – během zpracování požadavku webovou službou může z různých důvodů dojít k jejímu selhání. V závislosti na druhu chyby musí být mechanizmus zpracování výjimek schopen se s chybou vypořádat. Komplexní webové služby, které pro vykonání své činnosti používají dalších webových služeb, musí být schopny odstranit, vyřešit chyby vzniklé na straně těchto externích služeb. Jednou z možností, jak k chybě přistupovat, je po jejím výskytu počkat předem definovaný čas a znovu zaslat externí službě požadavek. Dojde-li opět k výskytu chyby, dostane se služba do chybového stavu. Jednou z možností, jak upozornit koncového uživatele na chybový stav služby je, že služba vrátí odpovídající chybový kód pomocí chybového portu. Alternativně mohou mít komplexní webové služby implementovaný mechanizmus pro kompenzaci neúspěšného volání externích služeb. Mechanizmus nebo lépe řečeno kompenzační strategie určuje, co se má stát v případě že volaná služba vrátí chybový kód nebo není dostupná. Podstatné je, že samotný kompenzační mechanizmus může reprezentovat další webové služby.
- *Souběžné zpracování* – Jelikož mohou komplexní webové služby využívat další webové služby, může být dokončení služby poměrně dost časově náročné. Jelikož může jedné službě trvat zpracování stejného dotazu v různý čas různě dlouhou dobu, není možné použít model synchronního volání služeb. Volající se může rozhodnout souběžně se službou spustit nějakou místní logiku, aby optimalizoval využití zdrojů nebo celkovou dobu běhu. Je důležité vědět, jak dlouho to které webové službě trvá.
- *Konkurentní datové vstupy a výstupy* – Pokaždé, když komplexní služba volá externí služby, musí mít k dispozici vstupní data, aby tyto externí služby mohly poskytnout odpověď. Může nastat případ, kdy nejsou vstupní

data v době volání externí služby známa. Musí tedy být možné poskytnout službě potřebná data za běhu, aby je mohla předat dál k externí službě.

- *Dynamická návaznost služeb* – Jak již bylo několikrát zmíněno, komplexní webová služba může volat další webové služby, což znamená, že musí mít implementován mechanizmus, který bude rozhodovat, jaká služba a kdy bude volána. Koncový uživatel služby nemá možnost ovlivnit, jaké služby budou volány, případně je nahradit jinými.

Důsledkem snadného přístupu k velkému množství informací na internetu je stále obtížnější hledání, prezentace a udržování informací, které uživatele potřebují. Mnoho iniciativ i komerčních subjektů se snažilo nalézt způsob, jak informace sémanticky obohatit tak, aby byly strojově čitelné a pochopitelné. Sémantický web, respektive sémantické webové služby jsou nástrojem, který poskytuje inteligentní přístup k heterogenním distribuovaným informacím. Umožňují tak softwarovým produktům zprostředkovat výměnu požadovaných informací mezi uživateli a zdroji informací.

6.1.1 Web Service Modeling Ontology

Potenciál pro dosažení dynamické, škálovatelné a efektivní infrastruktury pro elektronické transakce informací v podnikání a veřejné správě je poháněn rozvojem sémantických webových služeb, tedy služeb obohacených o strojově zpracovatelnou sémantiky (Roman, a další, 2005) a (WSMO, 2004). WSMO poskytuje konceptuální rámec a formální jazyk pro sémantický popis všech relevantních aspektů webových služeb s cílem usnadnit automatizaci rozpoznávání, kombinování a volání elektronických služeb prostřednictvím internetu. Základem pro WSMO byl v předchozí kapitole popisovaný WSMF, který byl očištěn od nadbytečných funkcionalit a rozšířen pomocí formální ontologie a jazyka (W3C WSMO, 2005).

WSMO poskytuje ontologické specifikace základních prvků sémantických webových služeb. Sémantické webové služby se zaměřují na integrované technologie nové generace internetu tím, že kombinují technologie sémantického webu a webových služeb. Obrací tak pohled na internet z pouhého úložiště informací na celosvětový systém pro distribuované výpočty. Proto je vhodné, aby rámce pro sémantické webové služby vycházely ze základních principů web designu, které jsou definovány pro vytváření sémantického webu, stejně jako ze zásad návrhu distribuovaných, servisně orientovaných způsobů výpočtu.

Jak uvádějí (Roman, a další, 2005) a (W3C WSMO, 2005), WSMO vychází z následujících principů návrhu:

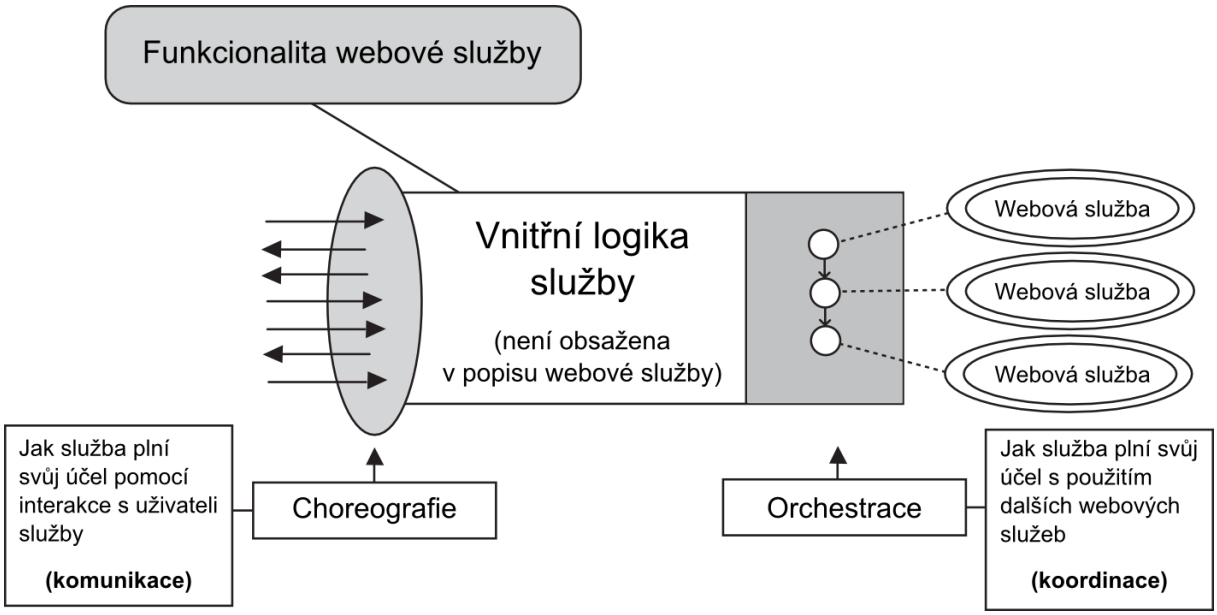
- *Shoda s webem* – WSMO dědí koncept URI (Universal Resource Identifier) pro unikátní identifikaci zdrojů jako základního návrhového principu webu. Navíc přijímá koncept jmenných prostorů pro označení konzistentních informačních prostorů, podporuje XML, decentralizaci zdrojů a další doporučení konsorcia World Wide Web Consortium (W3C).
- *Formální základ* – Ontologií se využívá jako datového modelu v celé WSMO, což znamená, že všechny popisy zdrojů, stejně jako všechna přenášená data jak uvnitř tak i mezi službami navzájem, mají formální základ. Ontologie jsou široce přijímány jako nejvyšší úroveň reprezentace znalostí, a proto byly zvoleny jako hlavní přístup pro sémantický web. Rozsáhlé využití ontologií umožňuje lepší zpracování informací a podporuje interoperabilitu.
- *Přísné oddělování* – Oddělování vyjadřuje, že zdroje WSMO jsou definovány izolovaně, což znamená, že každý zdroj je specifikován nezávisle bez ohledu na možné použití nebo interakce s jinými zdroji, což je v souladu s otevřenou a distribuovanou povahou internetu.
- *Centrálnost adaptérů* – Centrálnost adaptérů je komplementární návrhový přístup pro striktní oddělení služeb. Adaptéry řeší problematiku heterogenity dat, protokolů nebo procesů, ke které přirozeně dochází v otevřeném prostředí webových služeb.
- *Ontologické oddělování rolí* – Uživatelé vystupují v různých kontextech, které se nemusejí shodovat s kontexty dostupných webových služeb. WSMO rozlišuje mezi přáními koncových uživatelů a možnostmi dostupných služeb.
- *Popis versus Implementace* – WSMO rozlišuje mezi popisem sémantických webových služeb (popis) a vykonávacími prostředky (implementace). První z nich vyžaduje stručný a úplný popisný rámec založený na příslušných formalismech s cílem poskytnout stručný, sémanticky správný popis. Implementace se soustředí na podporu jak stávajících, tak i nově vznikajících technologií pro vykonávání webových služeb. WSMO se zaměřuje na poskytování odpovídajícího ontologického popisného modelu.
- *Vykonávací sémantika* – Formální vykonávací sémantiky nebo referenční implementace jako WSMX či další systémy technicky realizující WSMO umožňují ověření WSMO specifikace.

- *Služby versus Webové služby* – Webová služba je výpočetní jednotka, která umožňuje svým uživatelům dosáhnout určitého cíle. Naproti tomu služba je konkrétní činnost respektive hodnota, poskytována službou po jejím spuštění. WSMO poskytuje prostředky pro popis webových služeb, které poskytují přístup ke službám.

Jelikož WSMO vychází z WSMF, obsahuje stejné čtyři hlavní prvky, tedy *ontologie*, *cíle*, *popis webových služeb* a *adaptéry*, s cílem definovat sémantické webové služby. WSMO je meta-modelem popisujícím veškeré aspekty související se sémantickými webovými službami. Pro definování meta-modelu je použit MOF (Meta Object Facility) (OMG MOF, 2012), který definuje technologii řízení, abstraktní jazyk a rámec pro specifikaci a konstrukci neutrálních meta-modelů.

6.1.2 Orchestrace a choreografie ve WSMO

Choreografie a orchestrace jsou součástí definice interface popisu služeb ve WSMO. Popisují chování služeb ze dvou pohledů. Z pohledu komunikace, tedy jak komunikovat se službou jakožto s poskytovatelem informací a z pohledu spolupráce, neboli jak služba spolupracuje s ostatními službami definovanými v rámci WSMO a tím je jí umožněno poskytovat data nebo obecně informace.



Obr. 6.1. Obecný popis webové služby – vlastní zpracování autora podle (Roman, a další, 2005)

Jak vyplývá z obrázku 6.1, choreografie popisuje chování služby z pohledu svého uživatele. Orchestrace WSMO služeb definuje, jak je možné dosáhnout celkové funkčnosti služby prostřednictvím spolupráce s dalšími poskytovateli WSMO služeb. Popisuje, jak služba funguje z pohledu poskytovatele dalších služeb, které

daná služba využívá. Stavově orientovaný mechanizmus využívaný WSMO pro popis orchestrace a choreografie je založený na ASM (Abstract State Machine).

Jako základní model pro choreografii a orchestraci ve WSMO byl ASM zvolen z důvodů (Roman, a další, 2005):

- *Minimality* – ASM poskytuje minimální množinu modelovacích primitiv, tedy uplatňuje minimální ontologické závazky.
- *Maximality* – ASM je natolik popisný, aby jím bylo možné modelovat veškeré aspekty týkající se výpočtu
- *Formality* – ASM poskytuje pevný matematický rámec pro vyjádření dynamiky

7 Vybrané jazyky pro popis choreografie a orchestrace

Existuje mnoho standardů pro definici kooperace webových služeb, jako například WS-CDL, WSDL a WSCL, které popisují choreografie a BPEL4WS nebo XPDL, které umožňují popsat proces z pohledu orchestrace a také z pohledu chování rozhraní služeb. Pro popis struktury rozhraní se používá jazyk WSDL (W3C WSDL, 2001).

Jedním ze základních požadavků na WS-CDL je poskytnout nástroj pro verifikaci správného definování choreografie. Jak uvádí (Barros, a další, 2006), WS-CDL si sice vypůjčuje terminologii z pi-kalkulu či jiného podobného formálního popisného aparátu, ale neexistuje žádný postup komplexního mapování WS-CDL na pi-kalkul či jemu podobného formálního popisu. Obdobná situace je i v případě jazyka BPEL4WS, který ačkoli vychází z principů formálních matematických modelů, taktéž neobsahuje prvky pro verifikaci správného definování choreografie ani nástroje pro ověření sémantiky. Tento nedostatek kompenzují autoři, kteří na konkrétních případech ukazují možnosti použití formálních nástrojů pro verifikaci orchestrace. Jako formální nástroje pro verifikace používají konečné automaty (Fu, a další, 2004), Petriho sítě (Martens, 2004), (Verbeek, a další, 2005) nebo procesní algebru (Jing, a další, 2007).

Meta-model chování služby by měl být nezávislý na formátu vyměňovaných zpráv. Explicitně definovaný meta-model je základem pro definici formátu výměny zpráv a pro případné transformace modelu, což je důležité právě v případě modelování choreografie. Grafická reprezentace modelu choreografie a její následná transformace do XML je účinnějším postupem než modelovat přímo do XML. Jazyky používané pro popis orchestrace popisují interakce tak, že mohou probíhat pouze mezi dvěma účastníky, tedy jsou podporovány pouze bilaterální interakce.

Vztah mezi choreografií a abstraktními procesy v BPEL4WS nebo definicemi chování rozhraní v WC-CDL je netriviální a v současné době nejsou konkrétní představy o shodě mezi choreografiemi ve WS-CDL a abstraktními procesy v BPEL4WS (Barros, a další, 2006). Pochopení vzájemných vztahů je nezbytné, pokud potřebujeme použít oba přístupy společně. Definice vzájemných vztahů podobně jako mapování WS-CDL na BPEL4WS by bylo jednodušší, kdyby oba zmiňované jazyky měly podobnou sadu instrukcí pro řízení toku procesu. Základním rozdílem mezi konceptem choreografie a konceptem založeným na popisu chování rozhraní je, že choreografie nahlíží na interakce z globálního hlediska, zatímco popis chování rozhraní se zaměřuje vždy pouze na komunikaci

z pohledu jednoho z účastníků komunikace. Jazyky popsané v následujících kapitolách jsou založeny na XML a postrádají grafickou nadstavbu. Aby bylo dosaženo potřebného uživatelského komfortu při návrhu a verifikaci modelů choreografie a orchestrace, je vhodné využít některou z metod, která disponuje potřebným grafickým aparátem určeným pro popis takových modelů.

Mnoho autorů (Mendling a Hafner, 2008), (Peltz, 2003), (Barros, a další, 2006), (Busi, a další, 2006) a (Shapiro, 2002) zabývajících se problematikou modelování obchodních procesů, orchestrací a choreografií webových služeb shodně uvádí jazyky WS-CDL, WSDL, WSCL a nebo WSCI pro popis choreografie a jazyky BPEL4WS, XPDL pro popis orchestrace. Nicméně je zvláštní, že XPDL byl vyvinut pro zápis notace BPMN, která ve verzi 2.0 obsahuje element *Úloha v choreografii,* a tudíž by měl spíše spadat do kategorie jazyků pro popis choreografie.

7.1 Business Process Execution Language for Web Services

BPEL4WS definuje model a gramatiku pro popis chování obchodních procesů založených na interakci mezi procesy a partnery v podobě webových služeb. Interakce s partnery je realizována prostřednictvím rozhraní webových služeb. Model procesu v BPEL4WS definuje logiku a způsob jakým spolu jednotliví účastníci komunikují. Obsahuje mechanizmy pro ošetření výjimek a nestandardních situací. Hlavním účelem jazyka BPEL4WS je podpora automatizovaných vnitropodnikových a mezipodnikových obchodních procesů založených na webových službách.

Obchodní procesy mohou být popsány dvěma způsoby. Spustitelný model obchodních procesů zachycuje skutečné chování účastníků procesu a jejich interakce. Naproti tomu obchodní protokoly popisují procesy z pohledu zpráv, které si jednotliví účastníci vyměňují, a nezabývají se vnitřním chováním procesu. Popisy procesů pro obchodní protokoly se nazývají abstraktní procesy. BPEL4WS je použitelný jak pro modelování abstraktních, tak i spustitelných procesů. BPEL4WS je formální jazyk pro popis obchodních procesů a protokolů obchodních interakcí. Poskytuje nástroj, který usnadňuje integraci automatizovaných procesů (webových služeb) do vnitropodnikových i mezipodnikových obchodních procesů.

BPEL4WS využívá definici vlastností zpráv pro identifikaci relevantních dat vložených do zpráv. Na vlastnosti zpráv lze nahlížet jako na data, která buď přímo, nebo nepřímo ovlivňují chování obchodních protokolů. Komunikace probíhající uvnitř procesu ovlivňují nepřímo jeho chování navenek. Proces přímo ovlivňují data, která jsou přijímána zvenčí a jejichž obsah a struktura je předem

známa. Veškerá data, která přímo ovlivňují chování obchodních protokolů, musíme dle specifikace BPEL4WS považovat za vlastnost zprávy. Pro oba druhy dat zavádí (OASIS WSBPEL, 2007) pojmy transparentní a neprůhledná data.

Neprůhledná data představují transformace probíhající uvnitř procesu. Jejich projevem je nedeterministické chování procesu. Jelikož neznáme vnitřní strukturu, odvozujeme popis chování procesu podle jeho transparentních dat, tedy pozorovatelných výstupů. Přestože neznáme vnitřní rozhodovací mechanizmus, máme k dispozici množinu možných odpovědí na určité zprávy. Konkrétní výstup je nedeterministický, ale vždy bude roven jedné z předem známých hodnot, což umožňuje popsat chování procesu, aniž bychom znali konkrétní rozhodovací mechanizmus.

Definice obchodních protokolů je založena na specifikaci pozorovatelného chování a zpráv, které si vyměňují strany zapojené do protokolu. Protokol nepokrývá vnitřní procesy, na základě kterých jsou zprávy generovány. Důvody, pro které je vhodné oddělit pozorovatelné a vnitřní chování obchodních procesů, uvádí (OASIS WSBPEL, 2007). Firmy nechtějí dát k dispozici obchodním partnerům detaily vnitřního rozhodování v rámci procesů, jelikož se většinou jedná o jejich know-how. Na druhé straně je však nezbytné zabezpečit konzistenci v komunikaci i v případě, že se změní mechanizmus vnitřního rozhodování procesu.

Popis obchodních protokolů musí být nezávislý na konkrétní implementaci a zároveň musí pokrývat všechny aspekty komunikace jak v rámci organizace, tak i napříč organizacemi, které spolu komunikují. Na základě obchodního protokolu spolu mohou jednotliví účastníci komunikovat, aniž by byl nutný lidský zásah. BPEL4WS sám o sobě neumožňuje definovat osobu (člověka) jako účastníka procesu.

Obchodní protokoly nejsou vykonatelné a nepopisují toky uvnitř procesu. Prostřednictvím modelu vykonatelných procesů se popisují orchestrace. Choreografie služeb jsou popisovány pomocí abstraktních procesů. Podle (Peltz, 2003) rozlišuje BPEL4WS dva základní typy aktivit:

- *Jednoduchá aktivita* – je instrukce, která procesu umožňuje komunikovat s okolím. Příkladem jednoduché aktivity může být příjem a odesílání zpráv. Vykonatelný proces nejdříve přijme zprávu. Poté může následovat volání dalších služeb nebo jiná transformace dat a následně je odeslána reakce na zprávu.
- *Strukturovaná aktivita* – řídí průběh celého procesu a definují posloupnost volání jednotlivých webových služeb. Strukturované aktivity umožňují

vytvořit podmíněné smyčky a dynamické větvení. Tvoří základní programovací logiku BPEL4WS.

Neméně důležité prvky jsou proměnné a tzv. *partnerLinks*:

- *Proměnné* určují konkrétní data, která jsou vyměňována v rámci zasílání zpráv. Když proces přijme zprávu, vyplní příslušné proměnné tak, aby k nim měly přístup žádosti, které budou následovat. Proměnné se používají pro správu a zajištění perzistence dat napříč požadavky jednotlivých webových služeb.
- *PartnerLink* je odkaz na libovolnou službu, kterou proces vyvolává nebo na službu vyvolávající daný proces.

BPEL4WS disponuje robustním mechanizmem pro zpracování transakcí a výjimek. Pro seskupení více aktivit do jedné transakce se používá blok *scope*. Každá transakce má definovanou primární aktivitu, která představuje běžné chování procesu. Primární aktivitu může tvořit složitý proces s mnoha vnořenými a provázanými aktivitami. Transakce též může obsahovat bloky pro ošetření a kompenzaci chyb. Celý *scope* je sdílený se všemi aktivitami v rámci transakce. Každá z aktivit může využívat bloky pro ošetření a kompenzace chyb definované v tagu *scope*.

7.2 BPEL4People

Jak již bylo řečeno, jazyk BPEL4WS je bez dalších rozšíření primárně navržen pro podporu automatizovaných obchodních procesů založených na webových službách. Stejně tak jako bychom v praxi našli bezpočet scénářů, kde není zapotřebí interakce člověka v roli účastníka procesu, najdeme stejné množství případů, kde se proces bez interakce s lidmi v rolích vlastníků a účastníků procesu neobejde. Vývoj jazyka BPEL4WS se zaměřuje na dva hlavní směry.

Prvním z nich je model spustitelný procesů, který se používá ke specifikaci automatizovaných obchodních procesů, které koordinují – orchestrují činnosti webových služeb. Druhý směr se zaměřuje na pozorovatelné chování webových služeb. Obchodní procesy přesahují hranici pouhé orchestrace aktivit založených čistě na webových službách. Vyjma orchestrace webových služeb, procesy obvykle zahrnují jako další druh účastníků také lidi. Celkový model procesu je pak komplexní choreografií jak automatizovaných prvků, tak lidí, jejichž interakce nejen s automatizovanými účastníky, ale i mezi sebou je nutné správně modelovat. Neméně důležité je mít k dispozici odpovídající jazyk, kterým je možné popsat choreografie, které obsahují i lidi jako účastníky procesu. Bylo proto nutné definovat rozšíření jazyka BPEL4WS tak, aby umožňoval popsat

interakce lidí v rámci modelu orchestrace. BPEL4People je definován tak, že tvoří samostatnou vrstvu nad BPEL4WS tak, že je ho možné navázat na základní komponenty jazyka BPEL4WS, kdykoli je to zapotřebí.

Specifikace jazyka BPEL4People zavádí pro aktivity vykonávané lidmi nový typ aktivity, který umožňuje specifikovat interakce lidí a webových služeb v rámci procesů. Aktivity realizované lidskými účastníky mohou být definovány buď v rámci BPEL procesu nebo mohou být definovány samostatně pomocí specifikace WS-HumanTask (OASIS WS-HumanTask, 2010).

Aktivita vykonávaná lidmi je zvláštní typ aktivity přiřazený uživateli, vyžadující, aby uživatel vykonal nějakou činnost. Uživatel je prostřednictvím distribučního mechanizmu informován, že je od něho vyžadováno vykonání konkrétní činnosti. BPEL engine musí zpracovávat aktivity vykonávané lidmi odlišně od aktivit realizovaných webovými službami. Při spuštění aktivity vykonávané lidmi je vytvořena tzv. pracovní položka a ta je distribuována účastníkům procesu, kteří mají oprávnění aktivitu vykonat. Jeden z těchto oprávněných uživatelů k ní získá výhradní přístup. Pochopitelně může být místo více oprávněných uživatelů definován pouze jeden oprávněný uživatel, který začne aktivitu vykonávat. Jakmile uživatel vykoná potřebné činnosti, je aktivita ukončena a proces pokračuje ve zpracovávání dalších aktivit.

Každý účastník nebo skupina účastníků má definovanou roli, kterou v rámci procesu zastává. Použít lze jednu z následujících rolí:

- *Iniciátor procesu* – je osoba zodpovědná za vytvoření instance procesu a je určen na základě struktury procesu. Je také možné explicitně definovat konkrétního iniciátora procesu. Kompilátor poté musí zajistit, že alespoň jedna osoba bude asociována s touto rolí.
- *Účastník procesu* – je osoba nebo skupina osob, která může ovlivňovat průběh procesu. Účastníci procesu jsou asociováni s konkrétní instancí procesu. Pokud nejsou definování žádní účastníci procesu, iniciátor procesu se automaticky stává také účastníkem procesu. Obdobně jako v předchozím případě, i zde musí kompilátor zabezpečit, že bude alespoň jedna osoba definována jako účastník procesu.
- *Business Administrátor* – je osoba oprávněná provádět činnosti spojené se správou procesu, jako například řešit situace, kdy není dodržena doba určená pro trvání procesu. Na rozdíl od účastníků procesu jsou Business administrátoři vázáni na všechny instance konkrétního typu procesu.

Nejsou-li administrátoři definováni, přebírají jejich roli účastníci procesu. V rámci procesu musí být vždy alespoň jedna osoba asociována s touto rolí.

Každá aktivita vykonávaná lidmi má definovaného tzv. potenciálního majitele. Potenciální majitel může být jeden uživatel nebo skupina uživatelů, kteří mají právo proces započít a také dokončit. Vlastníkem procesu se vždy stane jeden z potenciálních vlastníků, když vytvoří instanci aktivity a začne ji vykonávat.

7.3 XML Process Definition Language

XPDL je jazyk pro výměnu definic procesů mezi různými produkty implementující procesní workflow, navržený skupinou Workflow Management Coalition (WfMC). Účelem XPDL je poskytnout univerzální jazyk umožňující přenášet procesní modely mezi různými systémy počínaje workflow management systémy, přes nástroje pro modelování procesů až pro simulační nástroje. Základní myšlenkou XPDL je použití minimální množiny konstrukcí pro definici procesů, která je obsažena ve většině prostředků pro tvorbu workflow. Naneštěstí tuto základní množinu konstruktů reflektuje jen malé množství dnes používaných workflow modelů, s nimiž se setkáváme v praxi (van der Aalst, 2003).

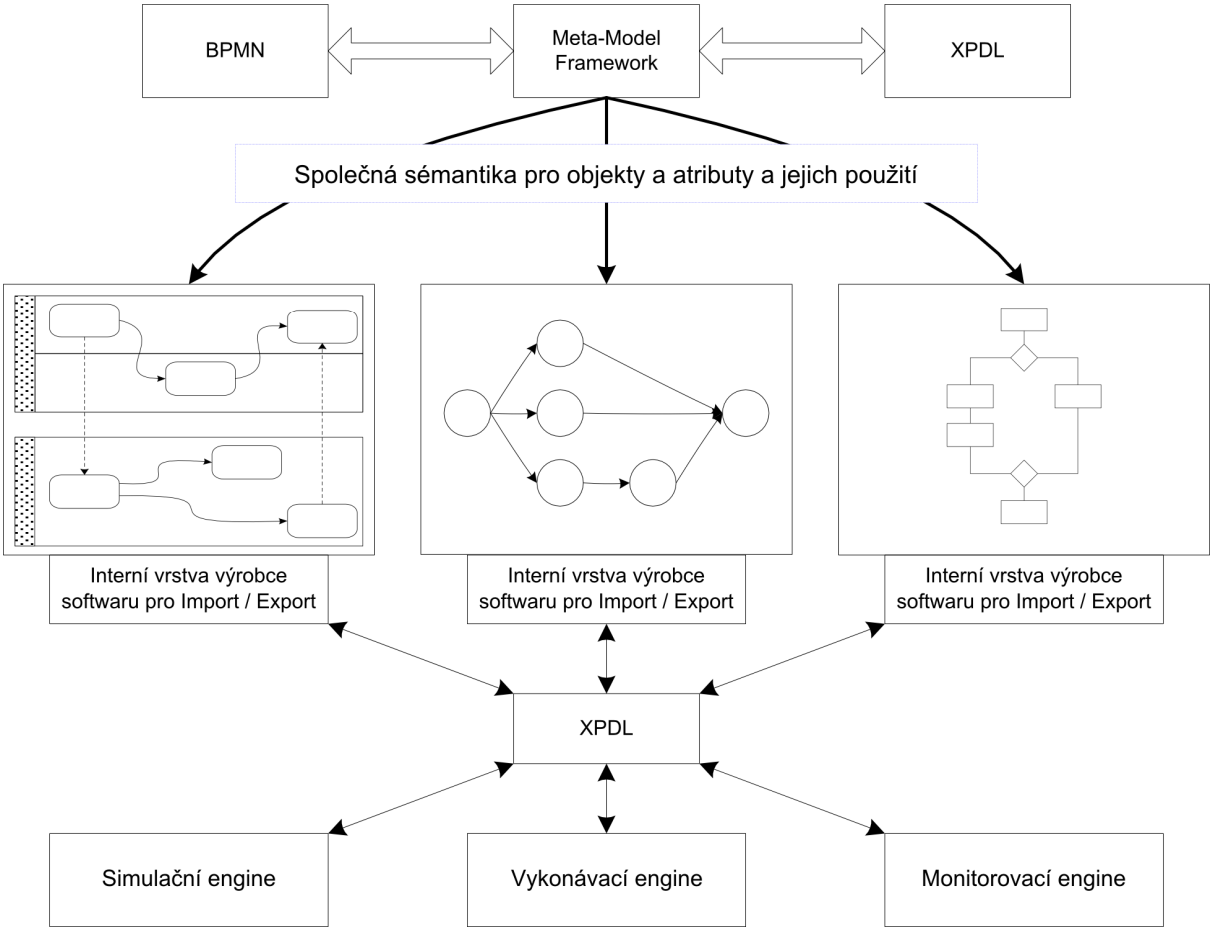
Jak již název napovídá, jazyk XPDL je založený na notaci XML. Hlavními prvky jazyka, jak je uvádí (van der Aalst, 2003) a (WFMC XPDL, 2008) jsou:

- *Package* – plní funkci kontejneru nebo obalu, který obsahuje ostatní prvky. Odpovídá procesnímu digramu notace BPMN.
- *Application* – obsahuje seznam aplikací, služeb nebo nástrojů vyžadovaných a používaných procesy, které jsou definované v rámci *Package*.
- *Workflow-Process* – se používá k definování workflow procesů nebo jejich částí. Prvek *Workflow-Process* se skládá z prvků typu *Activity* a *Transition*.
- *Activity* – je základním stavebním kamenem definice workflow procesu. Jednotlivé prvky typu *Activity* jsou spojeny pomocí prvků *Transition*. XPDL definuje tři typy aktivit – *Route*, *Implementation* a *BlockActivity*.
    - *RouteActivity* – jsou pouze fiktivní aktivity, které se používají pro směrování toku zpracování procesu.
    - *BlockActivity* – jsou aktivity, které se používají pro spouštění skupin rozsahem menších aktivit.

  - *ActivitySet* – blok na který odkazuje *BlockActivity*. Obsahuje odkaz na skupinu aktivit a přechodů.
  - *Implementation* – definuje jednotlivé kroky procesu, které jsou realizované manuálně, realizované jednou nebo více aplikacemi, anebo realizované nějakým subprocesem.

- *Transition* – každý přechod má tři základní vlastnosti – *form-activity*, *to-activity*, a *conditions*, tedy podmínky, za kterých je přechod realizován. Přechod z jedné aktivity do druhé může být buď podmíněný (může zahrnovat podmínky, které jsou vyhodnoceny, a na jejich základě poté přechod realizován je nebo není), nebo nepodmíněný. Přechody v rámci procesu mohou vést k sekvenčnímu nebo paralelnímu zpracování aktivit.
- *Participant* – obsahuje popis zdrojů, které budou vykonávat jednotlivé aktivity v rámci procesu. Jednotlivé zdroje, které mohou být přiřazeny k výkonu určité aktivity, jsou uvedeny jako atribut dané aktivity. Definice účastníka (*Participant*) nemusí odkazovat pouze na osobu. Může odkazovat na skupinu lidí (účastníků), kteří mají pravomoci aktivity vykonávat nebo na webové služby či jiné automatizované systémy.
- DataField a DataType – se používají pro definování všech proměnných, které bude proces používat. Obvykle se jedná o data, která se používají pro definování podmínek nebo základní parametry procesů, které jsou předávány mezi aktivitami.

Je zde patrná podobnost mezi jazykem XPDL a notací BPMN. Na rozdíl od XPDL definuje BPMN pouze grafickou podobu procesů, ale neupravuje, jak tyto definice procesů ukládat. Jak již bylo řečeno XPDL je jazyk založený na XML umožňující popsat všechny aspekty procesů definovaných v BPMN. Kromě grafického popisu procesů (diagramů), také obsahuje informace umožňující procesy spustit. BPEL4WS je vykonávací jazyk navržený pro orchestraci webových služeb. Definuje pouze vykonatelné aspekty procesů za předpokladu, že se proces zabývá pouze webovými službami. BPEL4WS neumožňuje definovat grafickou podobu procesů, podprocesy a procesy, v nichž jsou účastníci pouze lidé. Oba přístupy tedy BPMN i XPDL řeší stejný problém, ale z různé perspektivy. XPDL poskytuje formát využitelný pro výměnu definic procesů napříč nástroji pro modelování. Naproti tomu BPMN poskytuje grafickou notaci pro záznam komplexních procesů, která je srozumitelná pro vlastníky a účastníky procesů a je možné na jejím základě procesy implementovat do konkrétního informačního systému.



Obr. 7.1. Schematické znázornění koncepce výměny modelů procesů v XPDL mezi různými systémy – vlastní zpracování autora podle (WFMC XPDL, 2008)

Pro přenos definic procesů mezi systémy může být použit libovolný mechanizmus. Definice procesů ale musejí být vždy konzistentní za použití stejné množiny objektů, vztahů, vazeb a atributů, které vycházejí ze základního konceptu XPDL (WFMC XPDL, 2008). Vytvářet, přenášet a načítat validní XPDL v různých nástrojích je relativně jednoduše řešitelné. Jelikož XPDL neřeší problém sémantiky, je často velmi složité model, který byl vytvořen v nástroji *A*, načíst a interpretovat nástrojem *B* tak, jak bylo původně zamýšleno (van der Aalst, 2003).

7.4 Web Services Conversation Language

Existují různé způsoby, jak popsat komunikaci na úrovni obchodních procesů. WSCL je jednoduchý jazyk pro popis komunikací. Zaměřuje se na modelování sekvenčního zpracování komunikací nebo událostí jedné služby nebo procesu. Snaží se rozšířit možnosti jazyků, které nedisponují potřebným aparátem pro definici choreografie. Popis komunikací v jazyce WSCL volí formální přístup založený na XML. Původně se předpokládalo, že jazyk WSCL bude rozšířen o atributy definující QOS (quality of services), transakce a složené konverzace,

nicméně se tak nestalo, jelikož jeho další vývoj přestal být ze strany W3C podporován. Nespornou výhodou jazyka WSCL je právě jeho jednoduchost a formální popis (Cover, 2002). WSCL určuje, jaké informace budou v rámci komunikace přenášeny a pořadí výměny těchto zpráv. Nespecifikuje však strukturu zpráv, a proto se většinou využívá ve spojení s WSDL.

W3C (W3C WSCL, 2002) definuje následující čtyři hlavní bloky jazyka WSCL:

- *Document type descriptions* – definuje, jaké typy dokumentů budou v průběhu komunikace předávány
- *Interactions* – definuje typ interakce na základě výměny dokumentů. WSCL podporuje pět typů interakcí mezi účastníky komunikace: *Send* (služba vysílá odchozí dokument), *Receive* (služba přijme příchozí dokument), *SendReceive* (služba vysílá odchozí dokument a pak očekává, že obdrží příchozí dokument v odpovědi), *ReceiveSend* (služba obdrží příchozí dokument a pak vysílá odchozí dokument), *Empty* neobsahuje žádné dokumenty, používá se pro označení začátku a konce komunikace.
- *Transitions* – definuje pořadí komunikací mezi účastníky procesu. Specifikuje zdroj a cíl komunikace, případně určují typ dokumentu nebo zprávy jako podmínku pro uskutečnění komunikace.
- *Conversations* – seznam všech prvků typu *Interactions* a *Transitions*, které tvoří komunikaci. Dále atribut obsahuje název komunikace a konkrétní akci, kterou komunikaci začíná a případně končí.

Komunikace je definována z pohledu jednoho z účastníků, většinou z pohledu příjemce zpráv. První interakcí mezi účastníky je typu *Receive* nebo *ReceiveSend*. Iniciátor může odvodit průběh konverzace z definice konverzace protistrany tím, že odvodí vlastní interakce typu *Send* a *SendReceive* od interakcí typu *Receive* a *ReceiveSend* protistrany a naopak. Dva účastníci mohou navzájem komunikovat, pokud jsou definice jejich interakcí vzájemně duální.

Během komunikace mohou nastat v zásadě dva druhy chyb:

- *Chyby týkající se obsahu* – nastanou v případě, kdy účastník komunikace obdrží správný typ dokumentu se správnou strukturou, ale se špatnými hodnotami. Ošetření takových chyb se realizuje pomocí obslužných rutin obchodní logiky a definic konverzace, kde jsou pro jednotlivé chyby předem definovány odpovědi respektive typy dokumentů, které jsou odeslány jako odpověď v případě výskytu chyby.

- *Chyby v komunikaci* – vznikají při přijetí dokumentu jiného typu, než jaký je očekáván nebo v případě špatného pořadí zasílaných zpráv. Řešení takovéhoto druhu chyby je nutné definovat v komunikačním protokolu. Protokol může mít předdefinované chybové zprávy. Dále musí protokol definovat, co se stane v případě výskytu chyby. Instance konverzace po přijetí neplatného typu dokumentu může pokračovat, nebo je ukončena. Obdobně je nutné stanovit, jaké bude chování v případě, že účastník konverzace nebude dlouhou dobu na zprávu odpovídat.

WSCL nespecifikuje, jak jsou interakce navázány na konkrétní komunikační protokol. Pro popis vazby na konkrétní protokol je nutné použít jazyk, který disponuje potřebnými konstrukty pro popis takové vazby. Překladem jazyka pro popis vazeb na protokol je WSDL.

7.5 Web Service Choreography Interface

Jazyk WSCI je založený na XML, který slouží k popisu pořadí výměny zpráv mezi webovými službami a ostatními účastníky choreografované komunikace. Předpokladem pro použití WSCI je, že všechny choreografované služby jsou v rámci jedné organizace. Nepředpokládá se použití pro oblast B2B komunikace, ale pro popis rozhraní interních služeb nebo procesů (W3C WSCI, 2002).

WSCI popisuje dynamické rozhraní webové služby, které se účastní výměny zpráv. Dynamického chování rozhraní je dosaženo opakovaným použitím operací, které jsou definované pro statické rozhraní, které popisuje WSDL. WSCI je určen pro použití ve spojení s jazyky, které poskytují statický popis rozhraní webových služeb. Ačkoli může WSCI spolupracovat s libovolným jazykem, který má stejné vlastnosti jako WSDL, velké množství autorů se věnuje právě kombinaci těchto dvou jazyků (Estublier, 2005). Na WSCI a WSDL můžeme nahlížet jako na rozhraní. WSDL popisuje statické rozhraní definující služby jako body zpracovávající zprávy. WSCI popisuje dynamické rozhraní tím, že poskytuje předpisy pro přesně definované výměny zpráv mezi jednotlivými operacemi. Spojení těchto dvou jazyků dovoluje modelovat složité choreografie s velkým množstvím operací. WSCI se nezabývá popisem činností probíhajících uvnitř procesu, ale popisuje chování webových služeb globálně, pomocí rozhraní orientovaného na zprávy. Chování je vyjádřeno časovými a logickými závislostmi mezi předávanými zprávami, vzájemnými posloupnostmi a závislostmi zpráv. Popis chování také poskytuje informace, jakým způsobem budou ošetřeny výjimky.

WSCI popisuje vzájemné závislosti mezi webovými službami tak, aby každý klient chápal, jak se službou v rámci daného procesu komunikovat a aby mohl

předvídat její chování kdykoli v průběhu vykonávání procesu. Možnost popsat dynamické rozhraní služby v kontextu konkrétního procesu umožňuje zaměřit se na webovou službu z pohledu role, kterou v rámci procesu zastává. Abychom mohli služby začlenit do choreografie, musíme mít informaci o tom, jaké operace mohou zprostředkovat a jak mohou být tyto operace použity v daném případě. Tyto informace získáme tak, že definujeme vrstvu nad používanými webovými službami, která bude popisovat požadované chování, které musejí dodržovat všechny zúčastněné služby nebo procesy.

WSCI splňuje následující požadavky, které jsou nezbytné pro choreografii komunikačně složitých a dlouhotrvajících procesů (W3C WSCI, 2002):

- *Choreografie zpráv* – popisuje pořadí zpráv, v rámci komunikace, ve kterém budou přijímán a odesílány. Dále určuje pravidla, na jejichž základě bude pořadí zpráv určeno a také kdy komunikace začne a kdy skončí. Nedefinuje vnitřní strukturu procesu, ale stanovuje, jaké zprávy je proces schopen zpracovat.
- *Transakce* – popisuje, které operace jsou prováděny v transakčním režimu. Informuje ostatní účastníky komunikace o možnosti realizovat tyto operace. Transakční funkce umožňuje webovým službám zapojit se do distribuované transakce s ostatními službami, se kterými interaguje.
- *Zpracování výjimek* – definuje, jak bude webová služba reagovat na nestandardní situace. Popisuje alternativní chování v případě výskytu výjimky. Nedefinuje mechanismus, podle kterého webová služba spravuje výjimky a poruchy.
- *Management více paralelních komunikací* – popisuje, zda a jak je schopna webová služba řídit několik paralelních konverzací se stejným partnerem nebo s různými partnery. Dále popisuje požadovaný vztah mezi částmi různých zpráv, které patří do stejné komunikace.
- *Konektory* – popisují, jak propojit komunikace různých webových služeb účastnících se stejné výměny zpráv
- *Funkční kontext* – popisuje, jak se stejné webové služby chovají v různých případech výměny zpráv
- *Dynamická účast* – Interface WSCI popisuje, jak vybírat služby dynamicky. Výběr je závislý na kritériích, která jsou určena za běhu procesu a definovaná v interface služby. Známe-li parametry použité pro identifikaci služby, můžeme odvodit, jaký bude mít vliv změna některého z parametrů na chování služby.

WSCI je deklarativní jazyk pro popis rozhraní webových služeb, který nemůže být sám o sobě spuštěn. Umožňuje popsat pozorovatelné chování služeb a definovat pravidla pro komunikaci se službou (Mendling, 2008). Jazyk umožňuje dostatečně přesně a jednoznačně popsat rozhraní tak, že všichni účastníci komunikace budou v každé fázi daného procesu vědět, které zprávy může nebo musí služba odesílat či přijímat. Pozorovatelné chování každé strany, která se účastní komunikace, je popsáno nezávisle na ostatních.

7.6 Web Services Description Language

WSDL je specifikace popisující webové služby společnou XML gramatiku, která vznikla za spolupráce firem IBM a Microsoft jako reakce na potřebu popsat komunikaci nějakým strukturovaným způsobem. Služby popisuje jako množiny koncových komunikačních bodů, které si mohou vzájemně vyměňovat zprávy prostřednictvím zvoleného síťového protokolu. Operace a zprávy jsou definovány obecně a poté navázány na konkrétní síťový protokol a formát zpráv. V jazyce WSDL je obecná definice koncových bodů a zpráv oddělena od jejich konkrétního umístění a formátu dat, což umožňuje znuvupoužitelnost těchto obecně definovaných komponent. Konkrétní protokol a formát dat pro konkrétní typ portu tvoří opakovaně použitelné vazby.

Jak uvádí (Weerawarana, a další, 2005), WSDL popisuje čtyři kritické části předávaných dat:

- Interface informace popisující všechny veřejně dostupné funkce
- Datový typ informací pro všechny zprávy žádostí a zprávy odpovědí
- Závazné informace o transportním protokolu, který má být použit
- Informace o adrese pro nalezení požadované služby

WSDL dále definuje společný mechanizmus vazeb, čehož se využívá k propojení konkrétního protokolu nebo datového formátu a abstraktních zpráv, operací nebo koncových bodů.

7.7 Web Services Choreography Description Language

Činnosti, které se týkají různých organizací nebo nezávislých procesů musí být, pokud je to vyžadováno, koordinovány a musí navzájem spolupracovat za účelem dosažení společného cíle. Aby byla spolupráce úspěšná, je nutné definovat konkrétní pravidla mezi všemi účastníky procesu nebo komunikace (Barker, a další, 2009). Obyčejně se taková pravidla definují v běžném jazyce, což poskytuje prostor pro možná nedorozumění a nekorektní chování procesů.

Formální a standardizovaný způsob jakým pravidla definovat nabízí jazyk WS-CDL. Jazykem WS-CDL je možné jednoznačně popsat komunikace nebo vzájemné interakce mezi jednotlivými účastníky procesu z globálního pohledu. Pomocí jazyka WS-CDL definujeme globální podmínky a omezení pro komunikaci, které jsou platné pro všechny účastníky procesu. Aby mohl jakýkoli model orchestrace optimálně pracovat, je nutné, aby se všechny zúčastněné strany shodly na podmínkách a omezeních vzájemné komunikace. Výhodou takového modelu je, že pokud se dohodnutá pravidla a omezení nezmění, je vždy zaručena interoperabilita jednotlivých účastníků. Majitelé firem většinou dovolují systémovým integrátorům nebo analytikům, kteří ve firmě optimalizují obchodní procesy, nastavit omezení, která jsou nezbytná pro nastavení choreografie. Hlavním cílem při tvorbě WS-CDL bylo definovat deklarativní jazyk založený na XML, který z globálního pohledu popisuje chování, pravidla a omezení komunikace všech účastníků procesu (Fredlund, 2006).

WS-CDL není popisný vykonávací jazyk ani jazyk pro implementaci. Vykonávací logika aplikace musí být implementována některým z jazyků jako XLANG, WSFL, WSBPEL, BPML, XPDL, JLS, C# nebo dalších jim podobných. WS-CDL je tedy závislý na konkrétní implementaci.

Cílem specifikace WS-CDL je podle (W3C WS-CDL, 2003) umožnit:

- *Znovupoužitelnost* – Stejný model choreografie je použitelný různými účastníky v různých kontextech a jiném aplikačním software
- *Spolupráce* – Choreografie definuje posloupnost zasílání zpráv mezi dvěma nebo více nezávislými účastníky procesu tím, že popisuje, jak by měli navzájem spolupracovat
- *Spolupráce více stran* – Choreografie může zahrnovat libovolné množství účastníků
- *Sémantiky* – Choreografie může obsahovat nejen strojově čitelné informace jako dokumentaci, ale i sémantiku pro všechny komponenty choreografie
- *Uspořadatelnost* – Stávající choreografie je možné mezi sebou kombinovat a tím tvořit nové choreografie, které mohou být znovu použity v jiném kontextu
- *Modularitu* – Choreografie mohou být definovány pomocí prvku, který umožňuje spojit více choreografií do jedné

- *Zpracování výjimek* – Choreografie mohou definovat, jakým způsobem se má postupovat v případě výskytu neobvyklých událostí, které nastanou při provádění choreografie
- *Transakčnost* – procesy nebo jejich účastníci mohou pracovat v transakčním režimu. Transakčnost je způsob, jak koordinovat výsledek složitých a dlouhotrvajících komunikací
- *Informační uspořádání* – dovoluje účastníkům procesu, kteří jsou součástí choreografie, vzájemně komunikovat a synchronizovat informace.
- *Informačně řízená spolupráce* – choreografie popisují, jakým způsobem účastníci procesu procházejí jednotlivými stavy komunikace, pomocí popisu vzájemně vyměňovaných informací, podmínek a omezení.
- *Specifikace uspořadatelnosti* – definuje možnost spolupráce nebo rozšíření o prvky dalších specifikací jako WS-Reliability, WS-Composite Application Framework (WS-CAF), WS-Security, Business Process Execution Language for WS (WS-BPEL) a další.

WS-CDL popisuje vzájemnou spolupráci mezi účastníky. Jejich spolupráce probíhá na základě předem definovaných pravidel a omezení. Model WS-CDL se skládá z následujících bloků, které tato pravidla a omezení definují (W3C WS-CDL, 2003):

- *Role Type* – definuje druh chování neboli roli účastníka v závislosti na jeho postavení v kontextu ostatních účastníků
- *Relationship Type* – definuje vzájemné povinnosti účastníků, které musí být dodrženy, aby mohli vzájemně spolupracovat
- *Participant Type* – sdružuje ty části chování účastníka, které musí být implementovány stejnou logickou jednotkou nebo organizací, tedy účastníkem choreografie
- *Variables* – obsahují informace o objektech, které spolu vzájemně spolupracují, například uvádí, jaké informace si účastníci vzájemně předávají nebo mohou obsahovat informace o rolích služeb účastnících se
- *Tokens* – odkazují na části proměnných (*Variables*). Jak *Variables* tak i *Tokens* mají typy, které definují strukturu toho, co Variables obsahují nebo kam Tokens odkazují.

- *Choreografie* – definují formy spolupráce mezi jednotlivými účastníky procesu:
    - *Choreography Life-line* – vyjadřuje stav spolupráce. Nejdříve je spolupráce
      v počátečním stavu neboli ve stavu navázání komunikace, dále účastníci vykonají své aktivity a nakonec buď spolupráce skončí předpokládaným způsobem, nebo dojde k nějaké neočekávané události.
    - *Choreography Exception Blocks* – specifikuje, jaké další akce by měly být provedeny, pokud spolupráce neskončí očekávaným způsobem
    - *Choreography Finalizer Blocks* – popisuje, jak určit další interakce, které by měly proběhnout ještě před dokončením spolupráce
- *Channels* – specifikuje, kde a jak bude probíhat výměna informací mezi účastníky komunikace
- *Work Units* – stanovuje omezení, která musejí být dodržena, aby pokračovalo zpracování v rámci aktuální choreografie
- *Activities and Ordering Structures* – *Aktivity* jsou základní komponenty, z nichž se skládá choreografie a které transformují data, tedy vykonávají nějakou konkrétní aktivitu. *Ordering Structures* kombinují aktivity s ostatními *Ordering Structures* do vnořených struktur, aby vyjádřily uspořádání podmínek, za kterých komunikace v rámci choreografie mohou probíhat.
- *Interaction Activity* – Interakce jsou spolu s aktivitami základními stavebními kameny choreografií
- *Semantics* – umožňuje vytváření popisů každého prvku modelu za účelem zachycení sémantiky

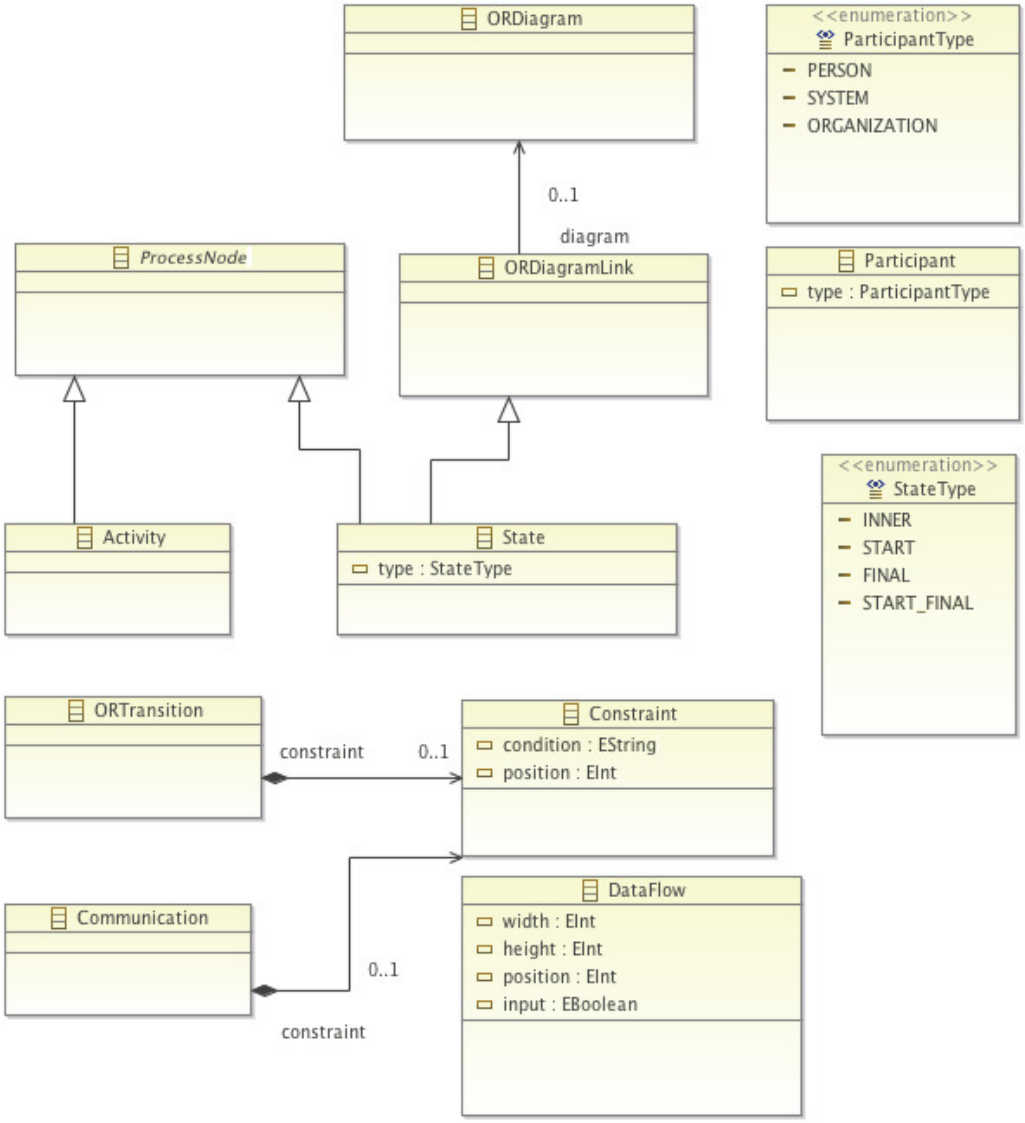
V případě WS-CDL jsou interakce popsány na bázi peer-to-peer komunikace. Účastníci konkrétní komunikace o sobě navzájem vědí, ale nevědí o ostatních službách, které jsou součástí komplexní komunikace. Vývojáři WS-CDL rozlišují mezi tradičními jazyky pro orchestraci, jakým je například BPEL4WS, který využívá již existující služby k orchestraci služby nové a mezi jazyky pro choreografii, jakým je WS-CDL, který popisuje interakce mezi existujícími službami, aniž by bylo nutné definovat služby nové. Vývojový tým jazyka WS-CDL uvádí, že je jazyk podložen procesním kalkulem respektive jedné z jeho

odnoží, a sice z pi-kalkulu. Jak se shodují (Fredlund, 2006) a (Barros, a další, 2006), spíš než realita je formální základ jazyka přáním otce myšlenky. Jsou zde pokusy, například (Yang Hongli, a další, 2006), o nalezení souvislostí a vazeb na pi-kalkul. Jazyk se setkal s širokou kritikou a není považován za příliš vhodný pro použití v praxi. Jako u většiny jazyků založených na XML, je obtížné a zdlouhavé zapisovat interakce, protože neobsahuje definice pro grafickou reprezentaci komunikací (Fredlund, 2006).

7.8 IZMAN CASE

Jedním z nástrojů implementujících metodu BORM je IZMAN Case, který byl vytvořen na Katedře informačního inženýrství na Provozně ekonomické fakultě ČZU v rámci projektu IZMAN. Důvodem, proč si autor této práce zvolil právě IZMAN Case, je spolupráce s jeho spoluautorem Ing. Robertem Perglem, PhD. na jeho dalším vývoji. Dalším důvodem je skutečnost, že zmiňovaný CASE nástroj je jedním z mála, který implementuje metodu BORM a byl vydán jako open-source, tedy je ho možné dále rozšiřovat. Dále autor předpokládá, že výsledky a závěry této práce bude možné využít pro další vylepšení tohoto modelovacího nástroje.

IZMAN Case je založen na Eclipse Modeling Framework (EMF), který využívá pro definici modelu jazyk Ecore. Vývojář může definovat model domény pomocí jazyka Ecore a automaticky je mu vygenerována kompletní implementace pro zvolený programovací jazyk, v případě IZMAN Case je to Java. EMF také dovoluje převést specifikace modelu z jiných jazyků, jako XSD nebo diagramy tříd jazyka UML na definici tříd jazyka Ecore (Steinberg, 2009). Jinými slovy, stačí sestavit model dané metody, v našem případě BORM například v UML jako diagram tříd, ve kterém pro každý prvek dané metody (participant, aktivita, stav, atd.) definujeme samostatnou třídu včetně atributů a případných závislostí mezi třídami. EMF pak vygeneruje implementaci vybrané metody nebo notace, na kterou je pak možné připojit například API pro manipulaci s jednotlivými prvky. Zároveň je automaticky upraven formát pro ukládání dat, aby odpovídal definovanému modelu.



Obr. 7.2. Ecore model ORD diagramu BORMu – poskytnuto vývojáři IZMAN Case

Zmiňovaný CASE nástroj umožňuje vytvářet procesní diagramy odpovídající metodě BORM, jak byla popsána v kapitole 4.3. Jak bude později v této práci zmíněno podrobněji, metoda BORM zatím nedisponuje potřebným konceptuálním ani grafickým aparátem, aby bylo možno její pomocí efektivně zaznamenat choreografii. Jazyk používaný pro záznam informací o procesních diagramech je založen na XML a nabízí se tedy přímé srovnání s podobným jazykem, který je taktéž založen na XML, tedy XPDL. Podobně jako XPDL (WFMC XPDL, 2008) obsahuje jazyk použitý v IZMAN Case elementy (tagy) pro definici:

- *Procesního diagramu* – v XPDL se jedná o tag <package>, jazyk který používá IZMAN Case používá pro celý diagram tag <diagrams> se specifikovaným typem *ORDiagram*. Oba slouží jako kontejner pro veškeré informace, které se procesu týkají.

- *Participantů* – Pro definici participantů využívá XPDL element vycházející z notace BPMN, tedy *Pool* naproti tomu jazyk popisovaného nástroje používá tag <blocks> typu *Participant*.
- *Stavů* – na rozdíl od BPMN a potažmo také XPDL, definuje BORM stavy, pro které je použit opět tag <blocks> typu *State.*
- *Aktivit* – oproti BORM, kde je definován pouze jeden typ aktivity (v tomto případě je pro aktivity použit tag <blocks> typu *Activity*), rozlišuje XPDL tři typy aktivit (*Route*, *Implementation* a *BlockActivity* – blíže popsáno v kapitole 7.3). Pro obecnou definici aktivity nicméně používá stejného tagu, a sice *Activity.*
- *Přechodů* – pro definici přechodů (mezi stavy a aktivitami jak je definováno v BORM) mezi aktivitami se používají tagy <transition> v XPDL respektive tagy <connectionsOut> typu *ORTransition* v IZMAN Case. Doplňující atributy jsou v podstatě shodné, tedy identifikace elementu, ze kterého přechod vychází a identifikace elementu kam je přechod směřován. Přechody, stejně jako komunikace, mohou být podmíněny. Pro zápis podmínky je využit tag <constraint>, který je vnořen to tagu <connectionsOut>. Obdobnou funkcionalitu poskytuje také XPDL v podobě tagu <condition>, který je vázán na konkrétní přechod.
- *Komunikací a datových toků* – Obecně pro zápis komunikace používá IZMAN Case opět tag <connectionsOut> typu *Commnunication*. Chceme-li přiřadit nějaké komunikaci datový tok, slouží k tomu tag <labels> umístěný v <connectionsOut>. Datové toky jsou typu *DataFlow*.
- *Podoby grafických symbolů* – Oba jazyky umožňují definovat velikost, umístění, barvy a tvar jednotlivých grafických prvků. Základní rozdíl je v tom, že XPDL byl vytvořen tak, aby byl schopen zaznamenat všechny aspekty notace BPMN, která se od BORM značně liší (blíže popsáno v kapitole 4).

Jak vyplývá z výčtu možností jazyka, který využívá pro záznam informaci o procesech IZMAN Case, tyto jsou dosti omezené a ve srovnání se zmiňovaným XPDL poskytují pouze nezbytné minimum proto, aby bylo možné diagramy ukládat. Srovnáme-li sadu grafických symbolů metody BORM a notace BPMN (blíže popsáno v kapitole 4), zjistíme, že BORM využívá nepoměrně menší množství symbolů a přesto je schopen popsat stejně složité situace jako BPMN. Nutno dodat, že množství symbolů v BPMN je dáno také tím, že obsahuje

konstrukty, které dovolují jednodušší převod do jazyka BPEL. Omezená sada symbolů tedy nemusí znamenat snížení vyjadřovací schopnosti.

Naneštěstí ani jazyk používaný v IZMAN Case ani zmiňovaný XPDL, které oba umožňují zaznamenat proces (s přihlédnutím ke specifikům jednotlivých notací, pro které byly vyvinuty) včetně informací o grafických prvcích, neumožňují popsat choreografii.

7.9 Komparace vybraných jazyků pro popis choreografie a orchestrace

Metoda BORM bez ohledu na nástroj, který ji implementuje, neumožňuje přehledným a jednoznačným způsobem zaznamenat choreografie a potažmo i orchestraci (Moravec a Brožek, 2012). Jelikož výše zmiňované jazyky bez ohledu na to, jak je jednotliví autoři klasifikují, umožňují zapsat především choreografii, lze se důvodně domnívat, že choreografie je obecnější pojem nežli orchestrace. Nicméně podrobněji se rozlišením pojmů orchestrace a choreografie budeme zabývat v dalších kapitolách, a to především v souvislosti s možnostmi, jak je zapsat pomocí metody BORM. Jak bylo popsáno v předchozí kapitole, jazyk používaný nástrojem IZMA Case neumožňuje přímo definovat choreografii případně orchestraci. Cílem je tedy nalézt takové konstrukty z jazyků, které požadované definice umožňují, o které by bylo možné jazyk pro záznam procesů popsaný v předchozí kapitole obohatit. Budeme vycházet z pravidel, která současný formát dat zavádí tedy, že jsou komunikace zaznamenány na straně odesílatele. Jsou tedy popisovány vždy pouze mezi dvěma participanty. Dále je nezbytné rozlišovat, o jaký druh dat se jedná, přestože je možné použít stávající konstrukty pro zápis datových toků. Ačkoli není nutné u každé komunikace specifikovat její pořadí, jelikož je proces zpracováván v rámci každého participanta sekvenčně, z pořadí komunikací lze usuzovat, zda se jedná o orchestraci či choreografii.

Základní kritéria pro rozlišení jednotlivých, výše popsaných jazyků, můžeme stanovit následující:

- Rozlišit, zda se jedná o iniciátora komunikace (procesu), nebo o ostatní účastníky. Zjednodušeně, zdali je možné rozlišit řídicí člen v rámci procesu.
- Zdali je možné rozlišit pořadí, ve kterém budou komunikace probíhat.
- Zda existuje možnost specifikovat data, která jsou součástí komunikace.

Následující tabulka 7.1 srovnává vybrané jazyky na základě stanovených kritérií.

	WSCL	WSCI+WSDL	WS-CDL	BPEL
Umožňuje rozlišit řídicí člen v rámci procesu	Ano, ale pouze nepřímo	ne	ano	ano
Umožňuje definovat pořadí komunikací	ano	ano	ano	ano
Lze specifikovat data	ano	ano	ano	ano
Jak popisuje komunikace	Z pohledu příjemce	Z globálního pohledu	Z globálního pohledu	Na bázi Peer-to-Peer

Tabulka 7.1. Srovnání vybraných jazyků pro zápis orchestrace a choreografie

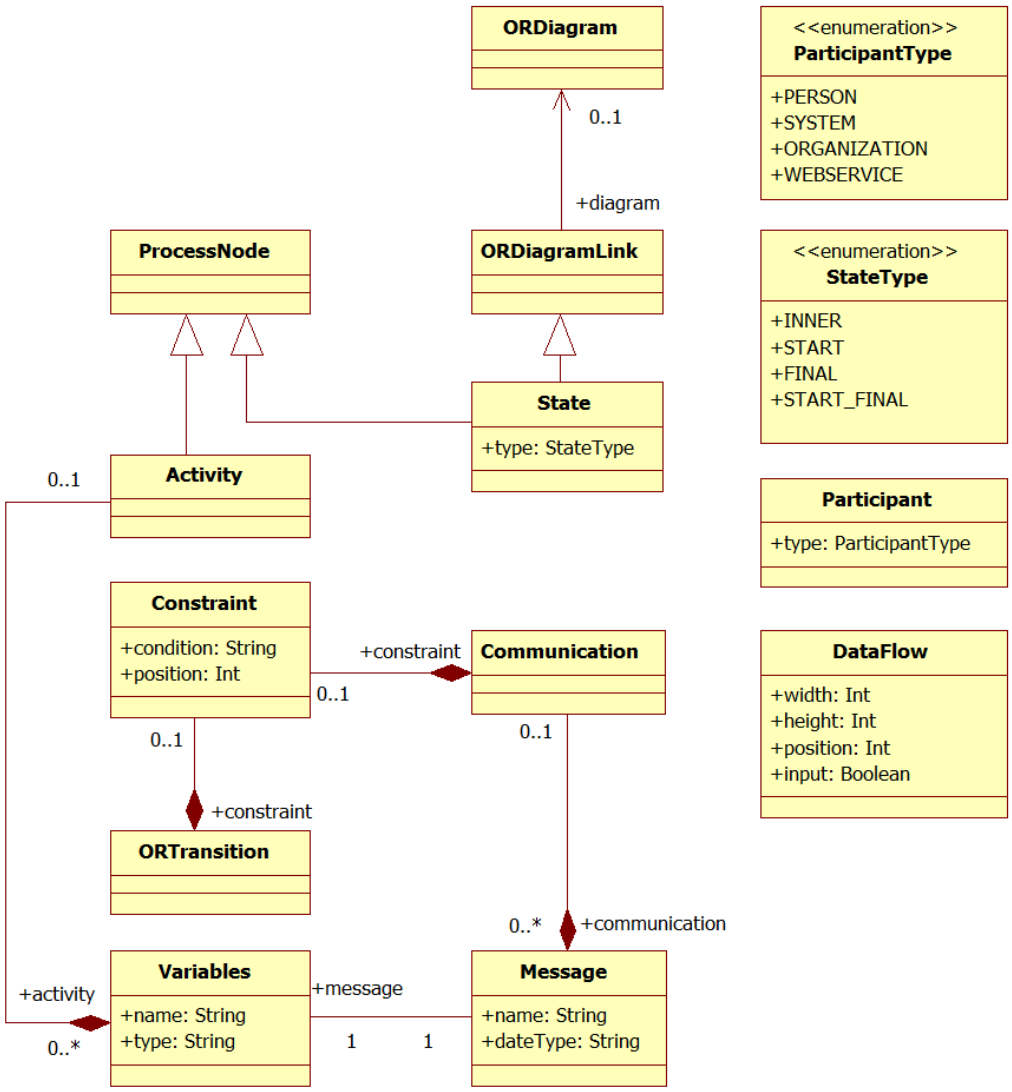
BORM definuje komunikace vždy mezi dvěma participanty, proto se jako vhodný koncept jeví použití vybraných konstruktů z BPEL (OASIS WSBPEL, 2007), což je souhrnné označení pro jazyky BPEL4WS a BPEL4People. Jelikož je BPEL úzce spjat s notací BPMN (Mendling a Hafner, 2008), lze se důvodně domnívat, že jeho rozvoj bude i nadále pokračovat stejně jako rozvoj BPMN, která je široce využívaná.

Navrhujeme rozšíření o nové konstrukty a upřesnění chápání stávajících prvků Ecore modelu metody BORM pro IZMAN Case tak, aby bylo možné lépe modelovat případy, kdy jsou participanty procesu webové služby. Navrhované změny se týkají zejména následujících oblastí:

- Strukturované aktivity v BPEL řídí průběh celého procesu a definují posloupnost volání jednotlivých webových služeb. Umožňují vytvořit podmíněné smyčky a dynamické větvení. Analogickou konstrukcí ke strukturovaným aktivitám jsou v BORM participanti. Není proto nezbytné zavádět nové konstrukce vycházející z BPEL, které by umožňovaly lépe řídit průběh procesu. Jednoduchá aktivita v BPEL je instrukce, která procesu umožňuje komunikovat s okolím. Příkladem jednoduché aktivity může být příjem a odesílání zpráv. Účastník procesu nejdříve přijme zprávu a poté může následovat volání dalších služeb nebo jiná transformace dat a následně je odeslána reakce na zprávu. Účel jednoduchých aktivit se překrývá jednak s funkcionalitou aktivit v BORM a také s případem, kdy je v jednom kroku procesu (alespoň z pohledu grafického záznamu) vykonáno více aktivit najednou. Stejně jako

v případě strukturovaných aktivit není nutné zavádět nové konstrukty a lze využít stávajících.

- Komunikace v BORM plní funkci propojení dvou participantů za účelem předávání řízení nebo zpráv. Obdobnou funkci, tedy propojení nebo lépe řečeno provázání dvou účastníků procesu, plní v BPEL *PartnerLinks* a není tedy nutné obdobnou funkcionalitu do modelu doplňovat.
- Pro definici obsahu zpráv využívá BPEL element *message*, který může obsahovat více než jednu proměnnou, přičemž každé proměnné přiřazuje kromě názvu také datový typ. Stávající podoba definice zpráv v IZMAN Case spočívá pouze v pojmenování datového toku, tedy ve slovním popisu dat, která by se měla v rámci komunikace navzájem předávat.
- BPEL používá proměnné k uchování zpráv, které jsou součástí stavu procesu. Uchovávané zprávy jsou většinou přijaty od ostatních účastníků procesu, nebo mají být ostatním odeslány. Proměnné také mohou obsahovat data nutná pro uchování informace o stavu procesu. Tato data nejsou nikdy předmětem komunikace. Jelikož v případě BORM dochází k výměně zpráv v rámci aktivit, jeví se jako vhodné řešení rozšířit aktivity o možnost definování proměnných podle vzoru BPEL, který používá element *variables.* Ke každé aktivitě by pak bylo možné definovat libovolný počet proměnných, používaných výhradně pro uchování informací o odesílaných a přijímaných datech, protože stavy procesu jsou v BORM, na rozdíl od BPEL, od aktivit oddělené.



Obr. 7.3. Rozšířený Ecore model BORMu – vlastní zpracování autora

Promítnutím výše navrhovaných úprav do původního Ecore modelu BORM, jak je prezentován na obrázku 7.2., získáme rozšířený model, jak je patrno z obrázku 7.3., který umožňuje lépe a komplexněji definovat komunikace mezi účastníky procesu. Zejména se toto rozšíření uplatnění v případě, kdy model procesu obsahuje webové služby jako participanty. Webové služby vyžadují vstupy splňující určitá kritéria. Jedná se například o specifické datové typy proměnných, jejich počet a podobně. Navrhovaná rozšíření by podle názoru autora této práce našla uplatnění i v případech, kdy proces nebude obsahovat webové služby jako účastníky procesu.

Změnou Ecore modelu by došlo i ke změně datového formátu, který je použit pro záznam informací o procesech. Případná implementace výše navrhovaného rozšíření by měla být předmětem dalšího rozšiřování nástroje IZMAN Case.

8 Orchestrace a choreografie

Choreografie vychází z modelu distribuovaného systému, který je řízen na základě předem definovaných pravidel bez centralizované kontroly. Orchestrace taktéž definuje pravidla komunikace v rámci distribuovaných systémů, ale je založena na centrálním řízení.

Základním rozdílem mezi orchestrací a choreografií z pohledu modelování procesů je přítomnost centrálního řídicího členu v případě orchestrace (Busi, a další, 2005). Orchestrace, myšleno orchestrované procesy, se také nazývají spustitelné procesy, protože jsou volány neboli spouštěny centrálním řídicím procesem – orchestrátorem. Orchestrátor zprostředkovává výměnu zpráv s ostatními účastníky procesu. Naproti tomu choreografie definuje náležitosti komunikace procesů nebo skupin procesů, které jsou na stejné úrovni a jsou stejně důležité. Choreografie můžeme zapsat pomocí orchestrací jednotlivých účastníků, kteří se na procesu podílejí. Na proces lze nahlížet podobně jako na skupinu webových služeb. Většiny procesů se účastní více participantů. Každý z účastníků vykonává nějaké činnosti-aktivity v závislosti na roli, kterou má v rámci procesu přiřazenou. Jednotliví účastníci mezi sebou komunikují, vyměňují si zprávy a předávají řízení procesu. Nelze jednoznačně říci, který proces bude řízen centrálně; určíme tedy jednoho z účastníků, který bude koordinovat činnosti ostatních a případ, kdy si budou řízení předávat jednotliví účastníci (Barros, a další, 2005). Použijeme-li analogii s koordinací webových služeb, narážíme na problém, kdy při řízení procesu nemůže řídicí člen v případě neobdržení odpovědi od některého z účastníků procesu tento výpadek kompenzovat dosazením nějaké předem definované odpovědi (Busi, a další, 2005) a (Barros, a další, 2006). Obchodní procesy v rámci organizací jsou založeny na vzájemné interakci mezi jejich účastníky (lidmi), proto nelze takové výpadky v komunikaci efektivně kompenzovat a je nutné vždy čekat na odpověď. Ve většině případů proto pro řízení procesů používáme kombinaci orchestrace a choreografie. Není tak významné, zda bude po nějaký čas průběhu procesu proces řízen centrálně a posléze tento centrální člen předá řízení někomu jinému a dál si budou řízení předávat další účastníci procesu.

8.1 Choreografie

W3C (W3C Glossary, 2004) definuje choreografii následovně:

> *„Choreografie definuje sekvenci a podmínky, podle nichž si více spolupracujících nezávislých agentů vyměňuje zprávy za účelem vykonání určitého úkolu vedoucího k cílovému stavu. Choreografie webových služeb se zabývá interakcí služeb a jejich koncových uživatelů. Každý uživatel webové služby, automatizovaný nebo reálný (člověk), je klientem této služby. Jinými slovy, uživatelé mohou být jak lidé, tak i jiné webové služby. Provádění transakcí mezi webovými službami a jejich klienty musí být jasně definováno. Jednotlivé transakce se mohou sestávat z více samostatných interakcí, jejich složení představuje kompletní transakce. Takové skládání transakcí, jejich protokolů pro výměnu zpráv, rozhraní, skládání jejich sekvencí a souvisejících logik považujeme za choreografii.“*[1]

Definice choreografie ve WSMO (WSMO Choreography in WSMO, 2005) vychází ze slovníku W3C:

> *„Choreografie popisuje chování z pohledu uživatele, což je v souladu s definicí obsaženou ve slovníku W3C. Tedy, že se choreografie webových služeb zabývá interakcí služeb s jejich uživateli.“*[2]

Myšlenka choreografie vychází z řízení posloupnosti komunikací procesů mezi různými organizacemi z globálního pohledu (Peltz, 2003). Z pohledu procesů se jedná například o interakce interních a externích účastníků. Model choreografie popisuje spolupráci a vzájemnou interakci účastníků procesu nebo služeb za účelem dosažení společného cíle. Choreografie nepopisuje akce probíhající uvnitř procesu, které nemají viditelný efekt navenek procesu, tedy popisuje pouze pozorovatelné chování. Obecně se jedná o transformace dat. Na interakce procesů pohlížíme z globální perspektivy, tedy se všemi účastníky zacházíme stejným

---

[1] A choreography defines the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state. Web Services Choreography concerns the interactions of services with their users. Any user of a Web service, automated or otherwise, is a client of that service. These users may, in turn, be other Web Services, applications or human beings. Transactions among Web Services and their clients must clearly be well defined at the time of their execution, and may consist of multiple separate interactions whose composition constitutes a complete transaction. This composition, its message protocols, interfaces, sequencing, and associated logic, is considered to be a choreography. (W3C Glossary, 2003)

[2] Choreography describes the behavior of the service from a user point of view; this definition is in accordance to the following definition given by W3C Glossary. Web Services Choreography concerns the interactions of services with their users. (WSMO Choreography in WSMO, 2005)

způsobem. Choreografie zachycuje všechny interakce a transformace dat mezi jednotlivými účastníky, které jsou významné pro průběh a výsledek sledovaného procesu. Zachycuje vzájemné závislosti, komunikace, datové toky, transakční závislosti a časová omezení účastníků procesu, atd. (Barros, a další, 2006) Choreografii můžeme podle (Mendling a Hafner, 2008) popsat z globální a lokální perspektivy. Globální pohled na choreografie pohlíží na výměnu zpráv jako na koordinaci komunikačních protokolů. Lokální perspektiva definuje vzájemné komunikace z pohledu konkrétního účastníka procesu. Lokální pohled označuje (Bussler, 2005) jako rozhraní procesu. Hovoříme-li o webových sužbách, popis komunikačního rozhraní může obsahovat základní interakce, typy zpráv, které může přijímat a omezující podmínky, za kterých odesílá nebo přijímá zprávy. Popis rozhraní také obsahuje popis závislostí na jiných službách (služba například nemůže být volána dříve, než přijde odpověď od služby, na kterou navazuje), tedy časová omezení, závislost na příchozích datových tocích a transakční závislosti (Barros, a další, 2006).

Lokální perspektiva se zaměřuje na popis rozhraní procesu pouze z jedné strany komunikace. Nelze ji proto považovat za úplný popis, jelikož každá komunikace musí probíhat mezi dvěma stranami. Komunikace z pohledu jednoho účastníka může mít více rozměrů podle toho, s kolika dalšími účastníky procesu komunikuje. Stejně jako globální pohled na choreografii ani pohled z lokální perspektivy nepopisuje nedůležité události uvnitř procesu.

Pro každou roli v rámci jedné choreografie může mít každý účastník definovaný libovolný počet definic chování rozhraní. Neboli může poskytovat stejné informace nebo služby, ale ne vždy musí být vnitřní chování procesu stejné.

Při modelování možných případů chování procesů, tedy při návrhu chování rozhraní v jednotlivých situacích, je nutné uvažovat nejen ideální případ, tedy že komunikace probíhá podle předem definovaných scénářů, ale především zachytit případy, kdy může při komunikaci dojít k nestandardnímu stavu. Vyskytnou-li se neočekávané nebo nestandardní situace, je nutné předem definovat podmínky, na jejichž základě i přes výskyt neočekávaného chování nedojde k uváznutí procesu. Hovoříme-li o interních účastnících procesu, můžeme jejich chování do jisté míry ovlivnit. Opačná situace je v případě externích účastníků procesu, kterých se modelování choreografií přímo týká. Ideální případ je stav, kdy jak předpokládané výstupy, tak i očekávané vstupy jsou v souladu s naším původním předpokladem. Praxe může přinést situaci, kdy dojde k výpadku externí služby. V takovém případě je odpovědný poskytovatel nebo zprostředkovatel služby. Nicméně je nutné možný výskyt takové situace předvídat a uzpůsobit vnitřní

strukturu procesu tak, aby případný výpadek komunikace neovlivnil další procesy nebo služby na takový proces navázané.

Pokud budeme na procesy nahlížet z obou zmiňovaných pohledů zároveň, můžeme říci, že pro každou ze stran, která se účastní komunikace, bude existovat jeden lokální model. Každý účastník procesu tedy bude mít vlastní rozhraní, které bude při komunikaci využívat. Jednotlivé lokální pohledy se nám spojí do globální choreografie, neboli budeme moci definovat jednotný komunikační protokol pro daný případ.

8.2 Orchestrace

Orchestrace podle W3C (W3C Glossary, 2004):

> *„Orchestrace definuje sekvenci a podmínky, za kterých jedna webová služba vyvolá další webové služby za účelem realizace nějaké funkčnosti. Neboli orchestrace je vzorem interakcí, podle kterého musí agent webové služby postupovat, aby dosáhl svého cíle.“*[1]

Orchestrace podle WSMO (WSMO Choreography in WSMO, 2005):

> *„Orchestrace webových služeb definuje, jakým způsobem dosáhnout požadované funkčnosti prostřednictvím poskytovatelů dalších služeb. Poskytuje popis funkce služby z pohledu poskytovatele, což je v souladu s definicí, kterou poskytuje W3C.“*[2]

Model orchestrace vychází z předpokladu, že v rámci komunikace mezi organizacemi existuje proces nebo spustitelná webová služba, která je poskytována pouze jednou ze stran. Tento spustitelný proces je někdy též označován jako integrační (Bussler, 2005). Zprostředkovává komunikaci jak mezi interními službami nebo účastníky procesu tak i s externími účastníky. Řízení komunikace s externími účastníky procesu přispívá také k úspěšné realizaci choreografie, ačkoli se jedná o lokální prvek procesu.

Model orchestrace popisuje jak komunikace s okolím, tak i události a transformace dat, které v procesu probíhají. Popis orchestrace může dále

---

[1] An orchestration defines the sequence and conditions in which one Web service invokes other Web services in order to realize some useful function. I.e., an orchestration is the pattern of interactions that a Web service agent must follow in order to achieve its goal. (W3C Glossary, 2004)

[2] The orchestration of a WSMO service defines how the overall functionality of the service is achieved by the cooperation of other WSMO service providers. It describes how the service works from the provider's perspective (i.e. how a service makes use of other WSMO services or goals in order to achieve its capability). This complies with the W3C definition of Web Service Orchestrations. (WSMO Choreography in WSMO, 2005)

obsahovat komunikace, podmíněné komunikace a závislosti mezi nimi. Popisuje i takové, které nejsou uvedeny jako součást rozhraní procesu. Důvodem, proč nejsou některé akce, případně vstupy a výstupy uvedeny ve specifikaci rozhranní, je, že rozhraní procesů nebo služeb jsou často k dispozici externím stranám a není nutné poskytovat rozhraní celé, ale pouze tu část, kterou bude externí strana využívat.

Pro popis orchestrace se používají především jazyky WSFL, BPML a BPEL, které jsou založeny na webových službách. Čím dál více se pro popis orchestrace využívá jazyk BPEL, což jednak odsouvá na pozadí obdobné jazyky, ale především se z jazyka BPEL stává standard pro popis orchestrace. Někteří autoři jako například (WSMO Choreography in WSMO, 2005) a (Schmidt a Reussner, 2002) se zabývají výzkumem obecných principů workflow managementu mezi organizacemi. Mnozí autoři (Barros, 2005) se zabývají jak standardy pro specifikaci choreografie, tak i vymezením formálních základů. Tradiční přístup k problematice choreografie, v kontextu definice závazných pravidel, spočívá v navržení obecného modelu, který by byl akceptovatelný napříč všemi stranami, které se choreografie účastní. Takový obecný model poté může být použit k odvození modelů lokálních choreografií pro každého z účastníků procesu. Každý z nich pak může realizovat vlastní orchestrovaný proces, který bude v souladu s celkovou choreografií (Mendling a Hafner, 2008).

Oba výše uvedené přístupy se překrývají, což je možné využít pro vzájemné ověření konzistence obou modelů. Jako výchozí bod pro vytvoření základního modelu orchestrace lze použít množinu předpokládaných chování procesů. Základní model lze poté rozšířit o detailní popis vnitřního chování jednotlivých procesů, čímž získáme komplexní model orchestrace. Model orchestrace, který pokrývá všechna předpokládaná chování procesu, je možné využít pro vytvoření alternativního modelu, který umožní procesu reagovat na nestandardní události. Alternativní model komunikace je nutné ověřit proti standardním a předpokládaným interakcím s ostatními účastníky procesu. Takovým přístupem je možné nalézt případy, kdy daný proces komunikuje s ostatními procesy v nesprávném pořadí, případně dojde k jeho uváznutí v důsledku čekání na interakci s jiným procesem, která ale nepřišla včas. Podobné případy chování je možné eliminovat buď přepracováním modelu orchestrace, nebo vložením dalšího členu mezi proces, respektive mezi jeho rozhraní a rozhraní ostatních procesů tak, aby transformoval nestandardní chování na chování, které je očekávané. Kompenzace výjimek tímto způsobem je možná, pokud můžeme předem říci, jaká má být odpověď nebo reakce na daný podnět (Peltz, 2003). Systémy založené na principu webových služeb, jako jsou účastníci procesu, mají

v této oblasti oproti ostatním účastníkům (lidem) výhodu. Mohou výjimky kompenzovat předem definovaným chováním, což z podstaty rolí osob v rámci procesů není možné. Ve většině případů je nutné čekat na konkrétní reakci osoby na událost. Obdobný způsob verifikace rozhraní a chování účastníků procesu je možné realizovat také v případně modelu choreografie (Alonso, a další, 2004).

9 Výchozí předpoklady disertační práce

Výchozí hypotéza byla formulována na základě prostudování vybraných zdrojů a zkušeností autora z projektů zabývajících se mapováním procesů na FIT na pražském ČVUT.

Jak uvádí (Merunka, 2008), procesní diagram v BORM je tvořen objekty (participanty), kde každý participant je modelován jako Mealyho konečný automat (popsaný v kapitole 5). BORM dále disponuje pokročilými nástroji pro verifikaci modelů a jejich transformace do modelů ekvivalentních k modelům podle MDA. Během studia literárních zdrojů autor nenalezl žádnou práci, ve které by byl procesní diagram BORMu jednoznačně formálním způsobem popsán pomocí konečného automatu.

Autor předpokládá, že současný trend spočívající ve využívání distribuovaných systémů a servisně orientované architektury bude i nadále pokračovat. Na metody pro modelování obchodních procesů budou kladeny stále vyšší nároky zejména v oblasti jednoznačného, přehledného a srozumitelného zachycení procesů, jejichž participanty mohou být také webové služby.

Autor této práce nenalezl žádný publikovaný výstup, který by se zabýval problematikou modelování orchestrace a choreografie procesů v BORM, vyjma (Moravec a Brožek, 2012), jehož je spoluautorem. Aby mohla tato metoda BORM obstát v konkurenci jiných, častěji používaných metod a aby ji bylo možné jednoduše používat také v případech modelování orchestrace a choreografie, jako je tomu například u BPMN, považuje autor za nezbytné a prospěšné jednoznačně definovat její formální základ.

Ačkoli je metoda BORM výrazně méně používaná, než například v této práci zmiňovaná BPMN, má podle názoru autora velký potenciál pramenící z výhod metody, jak je popisuje (Knott, a další, 2006). Toto tvrzení vychází především z vlastních zkušeností autora s projekty mimo akademickou sféru a dále také z projektů realizovaných při spolupráci s Fakultou Informačních technologií na pražské ČVUT, kde byla ve všech případech použita metoda BORM implementovaná nástrojem IZMAN Case.

Má-li být metoda BORM a potažmo i zmiňovaný CASE nástroj srovnatelný s ostatními přístupy a nástroji, musí následovat vývoj v oblasti integrace softwarových systémů. Jak vyplývá z předložené analýzy literárních pramenů, je zde patrná snaha o stále větší využívání webových služeb pro plnění úkolů v rámci obchodních procesů. Procesní diagramy sice umožňují zapsat

choreografie, ale jazyk využívaný pro jejich zápis toto explicitně nepodporuje, a proto by bylo žádoucí ho odpovídající způsobem rozšířit.

Případnou implementací výše uvedeného do modelovacího nástroje IZMAN Case, který je vyvíjen na KII, PEF, ČZU by bylo možné rozšířit možnosti jeho použití při modelování téměř libovolné problémové domény.

9.1 Motivace

Jedním z hlavních motivačních faktorů pro volbu právě tohoto zaměření disertační práce byla skutečnost, že se její autor věnoval využití metody BORM již v rámci své diplomové práce. Přestože se nejedná o masově používanou metodu, má své neoddiskutované klady, které jsou blíže popsané v kapitole 4.3. Nicméně metoda BORM má také omezení a nedostatky, z nichž některé by právě tato práce měla odstranit a posunout tak možnosti metody BORM.

Během svého doktorského studia se autor práce účastnil projektů vypsaných grantovými agenturami CIGA a IGA. Spolu s Ing. Martinem Papíkem, PhD byly zkoumány možnosti procesního modelování s pokročilejšími způsoby řízení (IGA 201011130009). Výzkum se mimo jiné věnoval možnosti využít Petriho sítě pro verifikaci a validaci procesních diagramů v BORM. Druhým projektem, na kterém se autor této práce podílel jako spoluřešitel, se zabýval znalostním modelováním životních situací v malých obcích pomocí přístupu modelování business procesů (CIGA 20101202). Jednalo se využití procesních diagramů pro zápis právních předpisů (zákonů, vyhlášek, atd.) tak, aby byly srozumitelné a lépe pochopitelné pro uživatele.

Mimo jiné, v rámci spolupráce s Ing. Robertem Perglem, PhD., který je kromě katedry informačního inženýrství na ČZU také členem Fakulty informačního inženýrství (FIT) na ČVUT, měl autor práce možnost využít metodu BORM v praxi. Praktické využití metody, které autorovi dovolilo odhalit případná slabá místa a nedostatky, probíhalo v rámci projektu zaměřeného na mapování procesů na FIT a něj navazujícího projektu, který se zaměřil na komparaci procesů mezi FIT a FEL na ČVUT.

Dalším motivujícím prvkem byla skutečnost, že se autor práce podílí spolu s Ing. Jiřím Brožkem, PhD. na vědecké činnosti související s rozvojem metody BORM, která vznikla Katedře informačního inženýrství Provozně ekonomické fakulty ČZU v Praze.

10 Formální popis ORD

Při volbě vhodného formálního aparátu pro popis procesních diagramů v BORM budeme vycházet z práce Merunky (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 a 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 rozdělit na více vzájemně propojených subprocesů. 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). Cílem této práce je ale popsat procesní diagram pomocí konečného automatu, a proto se dále nebudeme problematikou Petriho sítí zabývat.

10.1 Popis ORD pomocí konečného automatu

Jak již bylo řečeno, procesní diagramy v BORM, neboli ORD, 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 můžeme považovat za přechod, neboť neobsahuje žádná data. Naopak komunikace mezi participanty může obsahovat data, a proto je lze považovat za datové toky. Na celý procesní diagram lze proto nahlížet jako na množinu automatů, kde každý automat reprezentuje právě jednoho participanta.

10.1.1 Popis chování participanta

Vycházíme-li pouze ze základního konceptu konečného automatu, jak je popsán v kapitole 5, bude každý participant P_i reprezentující jedinečnou modelovanou entitu definován jako uspořádaná pětice $P_i\left(S_i, I_i, \delta_i, s_i^0, s_i^e\right)$, kde:

- 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^*. Dovolujeme čtení prázdného symbolu ϵ, takže participant může přejít do dalšího stavu, a to i bez zásahu uživatele.

Diskuse. V kapitole 5 byl formálním způsobem popsán konečný automat jako uspořádaná pětice $M(K, \Sigma, \delta, q_0, F)$. Na základě výše uvedeného aparátu můžeme nalézt ekvivalenci mezi popisem chování participanta $P_i\left(S_i, I_i, \delta_i, s_i^0, s_i^e\right)$ a popisem konečného automatu.

Konečný automat		Participant
K	$\leftrightarrow$	S_i
Σ	$\leftrightarrow$	I_i
$\delta : K \times \Sigma \to K$	$\leftrightarrow$	$\delta_i : I_i \times S_i \to S_i$
$q_0; q_0 \in K$	$\leftrightarrow$	$s_i^0; s_i^0 \in S_i$
$F; F \in K$	$\leftrightarrow$	$s_i^e; s_i^e \in S_i$

Tabulka 10.1. Ekvivalence popisu konečného automatu a popisu participanta

10.1.2 Proces s více nekomunikujícími participanty

Doposud jsme uvažovali model pouze s jedním participantem. Rozšiřme nyní náš model o N vzájemně nekomunikujících participantů $P_1,\ldots,P_N$, jejichž činnost budeme simulovat zároveň. Předpokládejme, že vstupní symboly jsou pro každého participanta odlišné.

$$\bigcap_{i=1}^{N} I_i \subseteq \{\varepsilon\}. \tag{10.1}$$

Můžeme definovat jeden konečný automat P_Σ, který bude kompozicí dílčích automatů reprezentujících jednotlivé participanty. Takový automat bude simulovat činnost všech participantů současně.

$$S = S_1 \times \ldots \times S_N,\ s_0 = \left(s_1^0, s_2^0, \ldots, s_N^0\right),\ s_e = \left(s_1^e, s_2^e, \ldots, s_N^e\right). \tag{10.2}$$

$$I = \bigcup_{i=1}^{N} I_i. \tag{10.3}$$

Potom vnitřní stavy P_Σ jsou n-tice o délce N, kde N je počet všech participantů, složené z vnitřních stavů jednotlivých participantů. Obdobným způsobem můžeme přistupovat k počátečním a koncovým stavům. Předpokládáme, 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.

$$\delta : I \times S \to S. \tag{10.4}$$

$$\delta\left(i, s_1, \ldots, s_N\right) = \left(s_1, \ldots, s_{j-1}, s'_j, s_{j+1}, \ldots, s_N\right) \mid \exists j \in \hat{N}, \delta_j\left(i, s_j\right) = s'_j. \tag{10.5}$$

Pokud bychom nepotřebovali zachytit vzájemné komunikace participantů procesu, mohli bychom považovat výše uvedený popis za úplný. Až doposud jsme pomocí konečných automatů popsali jednotlivé účastníky procesu a představili model procesu, který je tvořen jedním, komponovaným automatem složeným z dílčích automatů, z nichž každý odpovídá právě jednomu účastníkovi procesu. 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). 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. Naneštěstí zde již nevystačíme pouze se základní koncepcí konečného automatu. Koncept konečného automatu, který takové komunikace umožňuje, byl popsán v kapitole 5.3. Přítomnost komunikací vyžaduje významnou změnu oproti výše uvedené definici participantů, ale dovolí nám 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.

10.1.3 Proces s více vzájemně komunikujícími participanty

Syntézou formálního popisu Mealyho konečného automatu a vybraných prvků, které se týkají zasílání zpráv z formalizmu komunikujících konečných automatů (popsáno v kapitole 5.3) 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.

$$BP = \{\mathbb{P}^i\}. \tag{10.6}$$

Každého participanta pak můžeme popsat jako uspořádanou šestici $\mathbb{P}^i(\mathbb{S}^i,-\mathbb{M}^i,+\mathbb{M}^i,f^i,g^i,s_1^i)$, kde:

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

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

Diskuse. V kapitole 5.2 byl formálním způsobem popsán Mealyho konečný automat jako uspořádaná šestice $M(Q,p,\Sigma,\Delta,\delta,\mu)$. Na základě výše uvedeného aparátu můžeme nalézt ekvivalenci mezi popisem chování komunikujícího participanta $\mathbb{P}^i(\mathbb{S}^i,-\mathbb{M}^i,+\mathbb{M}^i,f^i,g^i,s_1^i)$ a popisem Mealyho konečného automatu. Tato ekvivalence umožňuje považovat každého participanta procesu za Mealyho konečný automat.

Mealyho konečný automat		Komunikující participant
Q	$\leftrightarrow$	$\mathbb{S}^i$
p	$\leftrightarrow$	$s_1^i; s_1^i \in \mathbb{S}^i$
Σ	$\leftrightarrow$	$+\mathbb{M}^i$
Δ	$\leftrightarrow$	$-\mathbb{M}^i$
$\delta: Q \times \Sigma \to Q$	$\leftrightarrow$	$f^i : \mathbb{S}^i \times +\mathbb{M}^i \to \mathbb{S}^i$
$\mu: Q \times \Sigma \to \Delta$	$\leftrightarrow$	$g^i : \mathbb{S}^i \times +\mathbb{M}^i \to -\mathbb{M}^i$

Tabulka 10.2. Ekvivalence popisu Mealyho konečného automatu a popisu komunikujícího participanta

Abychom mohli popsat celý procesní diagram v BORM neboli ORD pomocí jednoho komponovaného konečného automatu, musí být zajištěno, že mohou jednotliví participanti navzájem komunikovat a že každá komunikace může obsahovat data. Syntézou mechanizmu zasílání zpráv, který obsahují komunikující konečné automaty (jak je uvedeno v kapitole 5.3) a výše uvedeného můžeme za splnění určitých podmínek popsat mechanizmus zasílání zpráv v případě modifikované podoby Mealyho konečného automatu.

Definice 10.1. Předpokládejme, bez ztráty obecnosti, že participant nebude zasílat zprávu sám sobě.

$$-\mathbb{M}^i \cap +\mathbb{M}^i = \emptyset. \qquad (10.7)$$

Definice 10.2. Množina všech zpráv (komunikací) *i*-tého participanta pak bude sjednocením množin všech ochozích a příchozích zpráv.

$$\mathbb{M}^i = -\mathbb{M}^i \cup +\mathbb{M}^i. \qquad (10.8)$$

Definice 10.3. Bez ztráty obecnosti předpokládejme, že v celém systému komunikací je právě jedno identické m_i, tedy že každá zpráva má právě jednoho příjemce a jednoho vysílače.

$$\forall m_1 \in \mathbb{M}^1, m_2 \in \mathbb{M}^1 : m_1 \neq m_2. \qquad (10.9)$$

Definice 10.4. Množinu všech zpráv $\mathbb{M}^i$ tvoří uspořádané trojice $\langle \sigma^i, in^i, out^i \rangle$

$$\mathbb{M}^i = \{\langle \sigma^i, in^i, out^i \rangle\} = \{m\}^i, \text{ kde:} \quad (10.10)$$

σ^i je symbol

in^i, out^i jsou data. □

Definice 10.5. Každá zpráva má svého příjemce a odesílatele.

$$\forall \mathbb{P}^i : \bigcup_i -\mathbb{M}^i = \bigcup_i +\mathbb{M}^i. \quad \square(10.11)$$

Věta 10.1. Zaměřme se nyní na výše definovanou přechodovou funkci $g^i : \mathbb{S}^i \times +\mathbb{M}^i \to -\mathbb{M}^i$, jejíž sémantika je následující:

$\mathbb{P}^i$ je příjemce (*receiver*) zprávy a $\mathbb{P}^j$ je odesílatel (*sender*) zprávy.

Dále platí, že $m^{ij} \in +\mathbb{M}^i \wedge m^{ij} \in -\mathbb{M}^j$. □(10.12)

Věta 10.2. Pro příjemce zprávy pak můžeme definovat funkce $data(\mathbb{P}^i)$ a $in(m^i)$, pro které platí:

$$data(\mathbb{P}^i); in(m^i) = in^i; m^i = \langle \sigma^i, in^i, out^i \rangle. \quad (10.13)$$

$$in(m^i) = in(\langle \sigma^i, in^i, out^i \rangle) = m^i. \quad \square(10.14)$$

Věta 10.3. Analogicky definujeme funkce $data(\mathbb{P}^j)$ a $out(m^i)$ také pro odesílatele zprávy:

$$data(\mathbb{P}^j); out(m^i) = out^i; m^i = \langle \sigma^i, in^i, out^i \rangle. \quad (10.15)$$

$$out(m^i) = out(\langle \sigma^i, in^i, out^i \rangle) = m^i. \quad \square(10.16)$$

Přestože výměna zpráv probíhá z pohledu sémantiky BORMu v jednom okamžiku, aplikujeme-li teorii konečných automatů, musíme 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 10.4. Vzájemnou výměnu dat mezi participanty pak můžeme definovat:

pro příjemce jako:

$$data^{t+1}(\mathbb{P}^i) = data^t(\mathbb{P}^i) \cup in(m^{ij}) \wedge in(m^{ij}) \subseteq data^t(\mathbb{P}^j). \quad (10.17)$$

pro odesílatele jako:

$$data^{t+1}(\mathbb{P}^j) = data^t(\mathbb{P}^j) \cup out(m^{ij}) \wedge out(m^{ij}) \subseteq data^t(\mathbb{P}^i). \quad \square(10.18)$$

Pomocí formálního popisu vycházejícího z teorie konečných automatů jsme definovali účastníka procesu, který si může s ostatními participanty zasílat zprávy. Procesní diagram je tvořen množinou participantů, kterou získáme složením dílčích konečných automatů do jednoho komplexního automatu. Takový komplexní automat $\mathbb{FSM}^{BP}$ 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, 1991) nebo (Hopcroft, 2001), kteří popisují konkrétní algoritmy pro jejich skládání.

Na základě odvozené definice vzájemně komunikující participantů můžeme libovolný procesní diagram v BORM popsat jako konečný automat. Tento komponovaný automat bude tvořen množinou konečných automatů, z nichž každý bude reprezentovat právě jednoho participanta. Aplikujeme-li na množinu participantů algoritmus pro kompozici konečných automatů popisovaný v (Hopcroft, 2001), získáme $\mathbb{FSM}^{BP} = (\Sigma, \Phi, \Omega, \varphi, \gamma, \sigma_1) = \{\mathbb{P}^i\} = \{(\mathbb{S}^i, -\mathbb{M}^i, +\mathbb{M}^i, f^i, g^i, s_1^i)\}$, kde:

$$\begin{aligned} \Sigma &= \prod_i \mathbb{S}^i & \varphi &= \bigcup_i f^i \\ \Phi &= \bigcup_i -\mathbb{M}^i & \gamma &= \bigcup_i g^i \\ \Omega &= \bigcup_i +\mathbb{M}^i & \sigma_1 &= first\left(\prod_i s_1^i\right) \end{aligned} \qquad (10.19)$$

Jak je patrné z definice 10.19 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 způsob byla odvozena přechodová a výstupní funkce tohoto komponovaného automatu.

Přechodovou funkci pro tento komponovaný automat získáme s pomocí přechodových funkcí f^i dílčích automatů.

$$f : \Sigma \times \Omega \to \Sigma \qquad (10.20)$$

$$\begin{aligned} & f(Q^t, X) = Q^{t+1}; \\ & \left(\exists P^i \in BP : S_1, S_2 \in \Sigma(P^i)\right) \ \wedge \\ & \left(\exists\left(\left(S_1 \in Q^t\right) \wedge \left(X \in +\mathbb{M}^i(S_1)\right)\right) \ \wedge \ \exists\left(S_2 \in Q^{t+1}\right) \wedge S_2 = f^i(S_1, X)\right). \end{aligned} \qquad (10.21)$$

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

$$g : \Sigma \times \Omega \to \Phi \tag{10.22}$$

$$\begin{aligned} & f(Q, X) = X'; \\ & \left(\exists P^i \in BP : S \in \Sigma(P^i)\right) \quad \wedge \\ & \left(\exists\left((S \in Q) \wedge \left(X \in +\mathbb{M}^i(S)\right)\right) \wedge \exists\left(X' \in -\mathbb{M}^i(S)\right) \wedge X' = g^i(S, X)\right). \end{aligned} \tag{10.23}$$

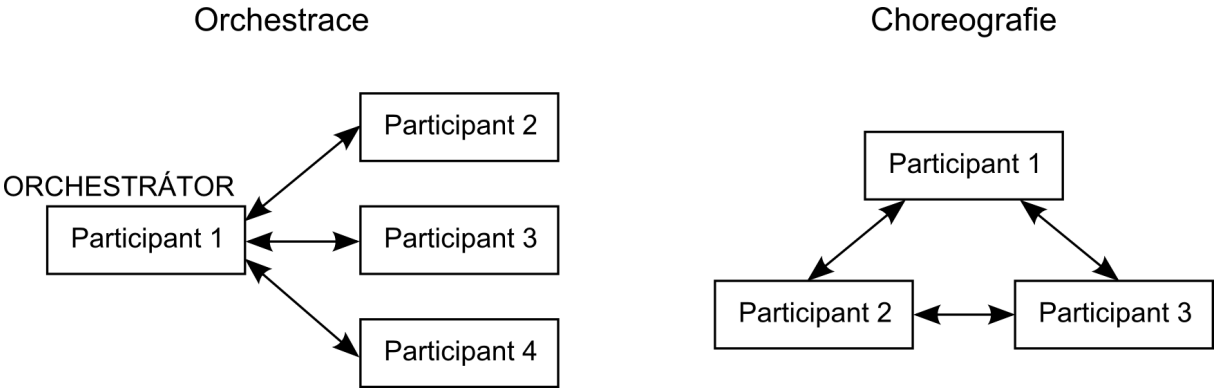
Jak jsme s pomocí výše uvedeného aparátu dokázali, lze popsat jakýkoli procesní model v BORM pomocí konečného automatu. Praktickým dopadem je možnost využití veškerých teoretických předpokladů a postupů, které jsou známy z oblasti teorie konečných automatů pro procesní modely v BORM. Lze tedy ověřit, 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.

11 Orchestrace a choreografie v BORM

11.1 Choreografie a orchestrace z pohledu BORM

Základním rozdílem mezi orchestrací a choreografií, jak je uvedeno v kapitole 8, je přítomnost řídicího členu v případě orchestrace. Jak vyplývá z rešerše jazyků pro popis orchestrace a choreografie v kapitole 7, je častěji v souvislosti s řízením komunikací uvažováno o choreografii. Obdobný závěr lze učinit při pohledu na grafické prvky, které využívají jednotlivé techniky pro popis procesů, představené v kapitole 4. Jelikož techniky modelování procesů uváděné v této práci obsahují prvky pro modelování choreografie a nikoli orchestrace, lze se důvodně domnívat, že choreografie za obecnější pojem nežli orchestrace. Pro formální popis orchestrace a choreografie využijeme aparátu, který byl definován v předchozí kapitole.



Obr. 11.1. Schématické znázornění rozdílu mezi orchestrací a choreografií – vlastní zpracování autora

Model orchestrace lze popsat jako proces, kde řídicí člen neboli *orchestrátor* je takový participant, jehož počáteční stav je zároveň počátečním stavem celého procesu a jeden z jeho koncových stavů je zároveň koncovým stavem celého procesu. Veškeré komunikace budou probíhat vždy mezi orchestrátorem a ostatními účastníky, nikoli však přímo mezi ostatními účastníky navzájem.

Věta 11.1. Mějme participanta P^i, který je orchestrátorem procesu BP a platí, že $P^i \in BP$. *Orchestrátor* je takový participant, pro kterého platí:

$$start(P^i) = start(BP) \quad \wedge \quad end(P^i) = end(BP). \qquad \square (11.1)$$

Definice 11.1. Funkce *start* a *end* pro participanta P^i můžeme definovat jako:

$$start(P^i) = start(\mathbb{S}^i, -\mathbb{M}^i, +\mathbb{M}^i, f^i, g^i, s_1^i) = s_1^i. \tag{11.2}$$

$$end(P^i) = end(\mathbb{S}^i, -\mathbb{M}^i, +\mathbb{M}^i, f^i, g^i, s_1^i) = \mathbb{S}_e^i, \tag{11.3}$$

kde $\mathbb{S}_e^i$ je množina všech koncových stavů participanta P^i a platí, že $\mathbb{S}_e^i \subseteq \mathbb{S}^i$. □

Definice 11.2. Analogicky pak definujme funkce *start* a *end* pro celý proces jako:

$$start(BP) = start(\Sigma, \Phi, \Omega, \varphi, \gamma, \sigma_1) = \sigma_1. \tag{11.4}$$

$$end(BP) = end(\Sigma, \Phi, \Omega, \varphi, \gamma, \sigma_1) = last\left(\prod_i \mathbb{S}_e^i\right), \tag{11.5}$$

přičemž funkce *last* definuje koncový stav celého procesu BP a platí, že $\left(\prod_i \mathbb{S}_e^i\right) \subset \Sigma$. □

Věta 11.2. Musí být splněna podmínka, že komunikace budou probíhat vždy mezi participantem P^i a ostatními účastníky $P^1, \ldots, P^n; n \neq i$, nikoli však přímo mezi ostatními účastníky navzájem. Dále budeme vycházet z definice 10.1., tedy že participant nezasílá zprávu sám sobě. Podmínku pak můžeme definovat jako:

$$+\mathbb{M}^i = \bigcup_i -\mathbb{M}^i \ \wedge \ -\mathbb{M}^i = \bigcup_i +\mathbb{M}^i, \tag{11.6}$$

dále platí

$$\bigcup_{j,k} in(m^{jk}) = \emptyset \ \wedge \ \bigcup_{j,k} out(m^{jk}) = \emptyset, \ j \neq k \neq i. \tag{11.7}$$ □

Model choreografie můžeme popsat jako proces, jehož počáteční stav bude jedním z počátečních stavů některého z participantů a jeho koncový stav je analogicky jedním z koncových stavů některého z participantů.

Věta 11.3. Mějme proces BP a množinu participantů $\mathbb{P}$ pro které platí, že $\mathbb{P} \in BP$. Dále platí:

$$start(BP) = start(\mathbb{P}) \ \wedge \ end(BP) = end(\mathbb{P}). \tag{11.7}$$ □

Definice 11.3. Funkce *start* pro množinu participantů $\mathbb{P}$ můžeme definovat jako jeden z počátečních stavů všech participantů, kteří se procesu účastní.

$$start(\mathbb{P}) = \prod_i start^i(P^i) = start^i(\mathbb{S}^i, -\mathbb{M}^i, +\mathbb{M}^i, f^i, g^i, s_1^i) = s_1^i. \tag{11.8}$$ □

Definice 11.4. Obdobným způsobem definujeme funkci *end* jako jeden z koncových stavů všech participantů, kteří se procesu účastní.

$$end(\mathbb{P}) = \prod_i end^i(P^i) = end^i(\mathbb{S}^i, -\mathbb{M}^i, +\mathbb{M}^i, f^i, g^i, s_1^i) = \mathbb{S}_e^i, \tag{11.9}$$

kde $\mathbb{S}_e^i$ je množina všech koncových stavů participanta P^i a platí, že $\mathbb{S}_e^i \subseteq \mathbb{S}^i$. □

Definice 11.5. Funkce *start* a *end* pro celý proces *BP* jsou analogické funkcím (11.4) a (11.5).

$$start(BP) = start(\Sigma, \Phi, \Omega, \varphi, \gamma, \sigma_1) = \sigma_1. \tag{11.10}$$

$$end(BP) = end(\Sigma, \Phi, \Omega, \varphi, \gamma, \sigma_1) = last\left(\prod_i \mathbb{S}_e^i\right). \tag{11.11}$$

Funkce *last* definuje koncový stav celého procesu *BP* jako jeden z množiny všech koncových stavů množiny participantů $\mathbb{P}$, za předpokladu, že $\left(\prod_i \mathbb{S}_e^i\right) \subset \Sigma$. □

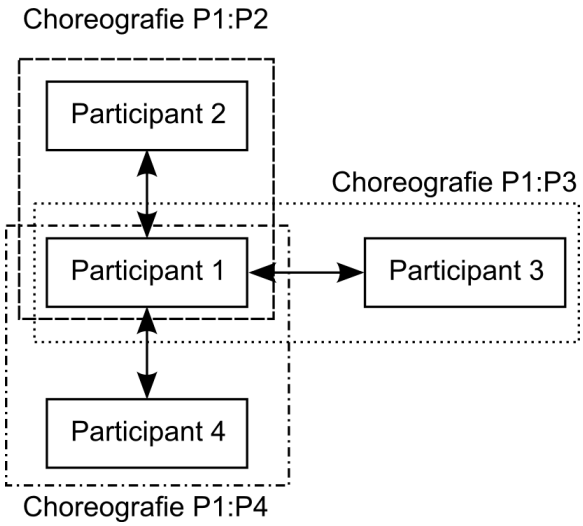
Srovnáme-li oba modely, zjistíme, že hlavním rozdílem je nutnost určení participanta, jehož počáteční stav bude zároveň počátečním stavem celého procesu. Obdobně je tomu u koncového stavu procesu, který je zároveň jedním z koncových stavů téhož participanta.

Budeme vycházet z předpokladu, že v rámci procesu, jak je popsán v BORM, jsou si všichni účastníci rovni. Každý proces je zahájen právě jedním participantem a koncový stav, ve kterém proces končí je jedním z množiny koncových stavů všech participantů, kteří se procesu účastní. Jednotliví účastníci procesu mezi sebou komunikují. Během realizace komunikace dochází k předávání zpráv a řízení procesu.

Zaveďme nyní pojem „minimální proces“. Minimálním procesem je myšlen takový proces, který obsahuje právě dva participanty, kteří mezi sebou komunikují. Bez dalšího jsou tito dva účastníci na stejné úrovni a oba mají stejnou důležitost. Minimální proces pak můžeme označit též jako minimální model choreografie, protože zde není definován řídicí člen a komunikace probíhá mezi nejmenším možným počtem účastníků. Budeme-li rozšiřovat minimální model procesu o další participanty, bude se tedy rozšiřovat i model choreografie.

Existují případy procesů, z jejichž povahy vyplývá přítomnost participanta, který bude koordinovat činnost ostatních účastníků. Jinými slovy, bude existovat takový participant, který bude celý proces zahajovat a v některém z jeho koncových stavů bude proces končit. Takový participant bude zahajovat komunikace s ostatními účastníky a bude určovat pořadí, ve kterém budou

komunikace probíhat. Participant bude vystupovat v roli *orchestrátora* procesu. Přestože je zde přítomen řídicí člen, v rámci dílčích komunikací budou dva aktuálně komunikující účastníci na stejné úrovni. Jak již bylo řečeno, komunikaci bude vždy zahajovat participant v roli *orchestrátora*. Zredukujeme nyní daný případ pouze na dva účastníky, kteří mezi sebou v daném okamžiku komunikují. Můžeme vyslovit předpoklad, že na úrovni právě dvou komunikujících participantů je rozdíl mezi orchestrací a choreografií pouze v tom, který z účastníků bude zahajovat komunikaci. Dále lze vyslovit tvrzení, že se v takovém případě se jedná o zvláštní případ modelu minimální choreografie. Rozšířením takové minimální podoby choreografie o informaci, který z participantů bude komunikaci zahajovat, získáme minimální model orchestrace.



Obr. 11.2. Schématické znázornění zápisu orchestrace pomocí choreografií – vlastní zpracování autora

Na základě výše uvedeného, lze vyslovit tvrzení, že orchestrace je zvláštním případem choreografie. Určíme-li jednoznačně iniciátora komunikací, lze jakýkoli model orchestrace zapsat pomocí takového počtu choreografií, který odpovídá počtu orchestrovaných participantů v původním modelu.

Diskuse. Přestože byly při formulaci formálního popisu choreografie stanoveny velmi striktní podmínky, bylo to nezbytné, protože v opačném případě by bylo značně obtížné rozlišit mezi orchestrací a choreografií. Ze zkušeností autora této práce z praxe je podstatná míra detailu neboli granulita, s jakou se k problému přistupuje při vytváření modelu procesu. Proces může být, při nižší úrovni detailu, podle definice a při splnění výše uvedených podmínek, považován za orchestraci. Naopak pokud bude stejný proces zachycen detailněji, může osahovat komunikace i mezi ostatními participanty a ne pouze komunikace mezi orchestrátorem a participanty. V takovém případě se již nejedná o orchestraci,

ale o choreografii, případně o kombinaci obojího. Jak vyplývá z výše uvedených definic, lze každý proces, který nesplňuje podmínky pro orchestraci považovat za choreografii.

Při vytváření procesního diagramu složitého procesu lze použít dva přístupy. Prvním z nich je postupovat shora dolů, tedy od větších a méně detailních celků, které se budou postupně rozpadat na detailnější podprocesy. Druhým přístupem je modelování problémové domény na nejnižší požadované úrovni a tyto detailní procesy spojovat do větších celků. V praxi se autor této práce setkal s oběma přístupy, nicméně vždy bylo úkolem procesního analytika vytvořit z detailnějších modelů modely s menší mírou detailu, a nebo naopak.

Během zpracovávání literární rešerše nenalez autor této práce žádnou práci, která by se zabývala automatizovanou agregací nebo dekompozicí procesních diagramů, které známe z metody BORM. Agregací a dekompozicí se v souvislosti s řízením zabývá například Adler (Adler, 1988), nicméně popisuje pouze dekompozice a agregace Data flow diagramů. Formální popis procesního diagramu v BORM otevírá možnost aplikovat algoritmy pro skládání a redukci konečných automatů jak je popisují (Gill, 1962), (Holzmann, 1991), a nebo například (Hopcroft, 2001). Skládáním a redukcí konečných automatů pomocí známých algoritmů a postupů lze z detailního procesního digramu automatizovaně vytvářet pomocí agregace méně detailní modely.

11.2 Procesy z pohledu participantů

Podle zkušeností autora této práce s modelování obchodních procesů z praxe je důležité v případě vedení interview, na jehož základě budou posléze vytvářeny procesní modely, rozlišit, jak chápou své role v rámci procesů vlastníci a jak je případně chápou jejich nadřízení a manažeři.

Ačkoli k nepochopení rozdílu mezi rolí a procesem dochází častěji u složitějších procesů, můžeme rozdílná pochopení rolí demonstrovat na jednoduchém příkladu procesu výměny pneumatik v autoservisu.

Mějme hypotetický proces, modelovaný z perspektivy autoservisu, kterého se účastní tři participanti – zákazník, mechanik a pracovník skladu. Celý proces je koncipován následovně. Zákazník přijíždí do autoservisu s požadavkem na výměnu opotřebovaných pneumatik s tím, že tyto hodná ponechat v autoservisu k likvidaci. Zákazníka se ujímá mechanik, který zjistí typ a parametry pneumatik, které bude pro výměnu potřebovat. Obrací se na pracovníka skladu s žádostí o čtyři kusy pneumatik majících patřičné parametry. Pracovník skladu

nalezne požadované pneumatiky a vydá je mechanikovi, který následně provede výměnu. Zákazník následně opouští autoservis s vyměněými pneumatikami.

Povedeme-li interview s mechanikem, pravděpodobně se nám dostane informace, že proces výměny pneumatik, kterého se účastní, má následující průběh – Přijíždí zákazník, zjistím jaké potřebuji pneumatiky, vyžádám si je ve skladu a provedu montáž. Naproti tomu, pro pracovníka skladu bude mít proces výměny pneumatik z jeho pohledu jiný průběh – Přichází mechanik a požaduje vyskladnění určitého typu pneumatik, jsou-li k dispozici, tak mu je vyskladním. Oba participanti popisovali nikoli dva různé procesy, ale pouze jeden proces výměny pneumatik, ale každý z nich vystupoval v jiné roli. Je proto nutné provádět interview nejen s konkrétními účastníky procesu, ale i s osobami majícími pravomoci definovat náplně jednotlivých procesů.

Pokud je proces složitější, pak samotný sled stavů a aktivit jednoho participanta je jeho role v rámci tohoto procesu. Naneštěstí je často role chápána, nejen účastníky procesu ale i lidmi zodpovědnými za definování náplně procesů, jako samostatný proces a nikoli jako role. Pravděpodobnou příčinou je nedostatečné pochopení širších souvislostí popisovaného problému. Se stejným nepochopením rozdílu mezi rolí v rámci procesu a procesem samotným se lze setkat také v prováděcích předpisech a příručkách pro manažery. Při vedení interview je tedy nutné důsledně rozlišovat mezi procesem a rolí konkrétního participanta v rámci procesu.

11.3 Srovnání BPMN a BORM

Následující kapitola je případovou studií, jejímž cílem je demonstrovat základní vlastnosti metody BORM pro popis procesu a dále ukázat hlavní rozdíly mezi procesním diagramem v BORM a BPMN. Na stejném příkladu procesu budou prezentovány možnosti BPMN a BORM pro modelování choreografie a potažmo i orchestrace.

11.3.1 Definice problémové domény

Problémovou doménou je Fakulta informačních technologií (FIT) na pražském ČVUT. Po vzniku FIT nebyly žádné procesy přesně definovány a prováděly se buď ad-hoc nebo podle toho, jak probíhají na ostatních fakultách ČVUT. Jelikož je každá fakulta více či méně specifická, nebylo možné beze zbytku převzít procesy z jiné fakulty. Nejprve probíhalo mapování procesů za pomoci metody BORM a následně, z důvodu potřeby porovnat získané modely s již existujícími modely procesů na Fakultě elektrotechniky (FEL), byly převedeny do BPMN. Notace BPMN byla zvolena z důvodu unifikace procesů s FEL, která má veškeré

modely právě v této notaci a také proto, že srovnání napříč metodami by bylo časově náročnější. Model procesu, který bude dále prezentován, byl sestaven na základě podkladů získaných během dvou interview vedených se zástupci FIT.

11.3.2Zobrazení procesu v BPMN a BORM

Na zvolenou problémovou doménu byly aplikovány všechny kroky OBA pro analýzu procesního modelu podle BORMu, jak je uvádí (Merunka, 2008), ale jelikož práce není primárně zaměřená na popis postupu modelování této konkrétní problémové domény, budou zde uvedeny pouze segmenty odpovídající zvolenému procesu.

Jak je uvedeno v kapitole 4.3.2, skládá se OBA z pěti kroků. Kvantifikaci výstupů jednotlivých kroků v rámci analýzy celé problémové domény uvádí následující tabulka 11.1.

Krok OBA	Počet
Rozpoznané procesy	63
Identifikace participantů	52 (z toho 7 externích)
Definované scénáře	63
Modelové karty	52
Procesní diagramy	69

Tabulka 11.1. Přehled výstupů OBA v rámci mapování procesů na FIT

Jednotlivé kroky OBA budou demonstrovány na procesu „Předložení žádosti o vydání publikace ediční radě ČVUT“.

Rozpoznání procesů

Pomocí řízeného interview byl mimo jiné identifikován proces „Předložení žádosti o vydání publikace ediční radě ČVUT“, jehož slovní popis následuje.

Proces pokrývá agendu spojenou s předložením žádosti o vydání publikace ediční radě ČVUT, která se koná dvakrát do roka. Za FIT předkládá žádosti Ediční referent. Ediční rada pak rozhoduje, zda publikaci schválí k vydání či nikoli. Iniciátorem procesu je Ediční referent, který dvakrát do roka předkládá ediční radě návrhy na vydání publikací. Ediční rada pak návrh posoudí, a buď schválí, nebo neschválí. Ediční referent následně uvědomí zaměstnance, který má v úmyslu vydat publikaci, o schválení či neschválení Ediční radou a dalším možném postupu. Výsledkem je buď schválená, nebo nechválená žádost o vydání publikace.

Identifikace participantů

Na základě verbálního popisu byli identifikováni následující účastníci procesu (participanti):

- *Zaměstnanec* – libovolný zaměstnanec FIT, který by mohl chtít vydat publikaci
- *Ediční referent FIT*
- *Ediční rada ČVUT*

Definice scénářů

Jednotlivým rozpoznaným procesům definujeme odpovídající scénáře. Pro identifikaci procesů sloužil verbální popis, který je po převedení do podoby scénářů mnohem lépe strukturován. Kromě třech základních fází procesu (iniciace, akce, výsledku procesu) obsahují scénáře informaci o tom, kteří participanti se procesu účastní a v jaké roli.

Předložení žádosti o vydání publikace ediční radě ČVUT	
initiation: Ediční referent FIT předkládá žádosti ediční radě ČVUT	roles: Ediční referent FIT - initiates Ediční rada ČVUT - approves Zaměstnanec - cooperates
action: Posouzení předložených návrhů ediční radou ČVUT	
result: Referent informuje žadatele o schválení/neschválení jejich návrhu	
Derived diagrams: Předložení žádosti o vydání publikace ediční radě ČVUT	

Obr. 11.3. Scénář procesu „Předložení žádosti o vydání publikace ediční radě ČVUT“

Modelové karty

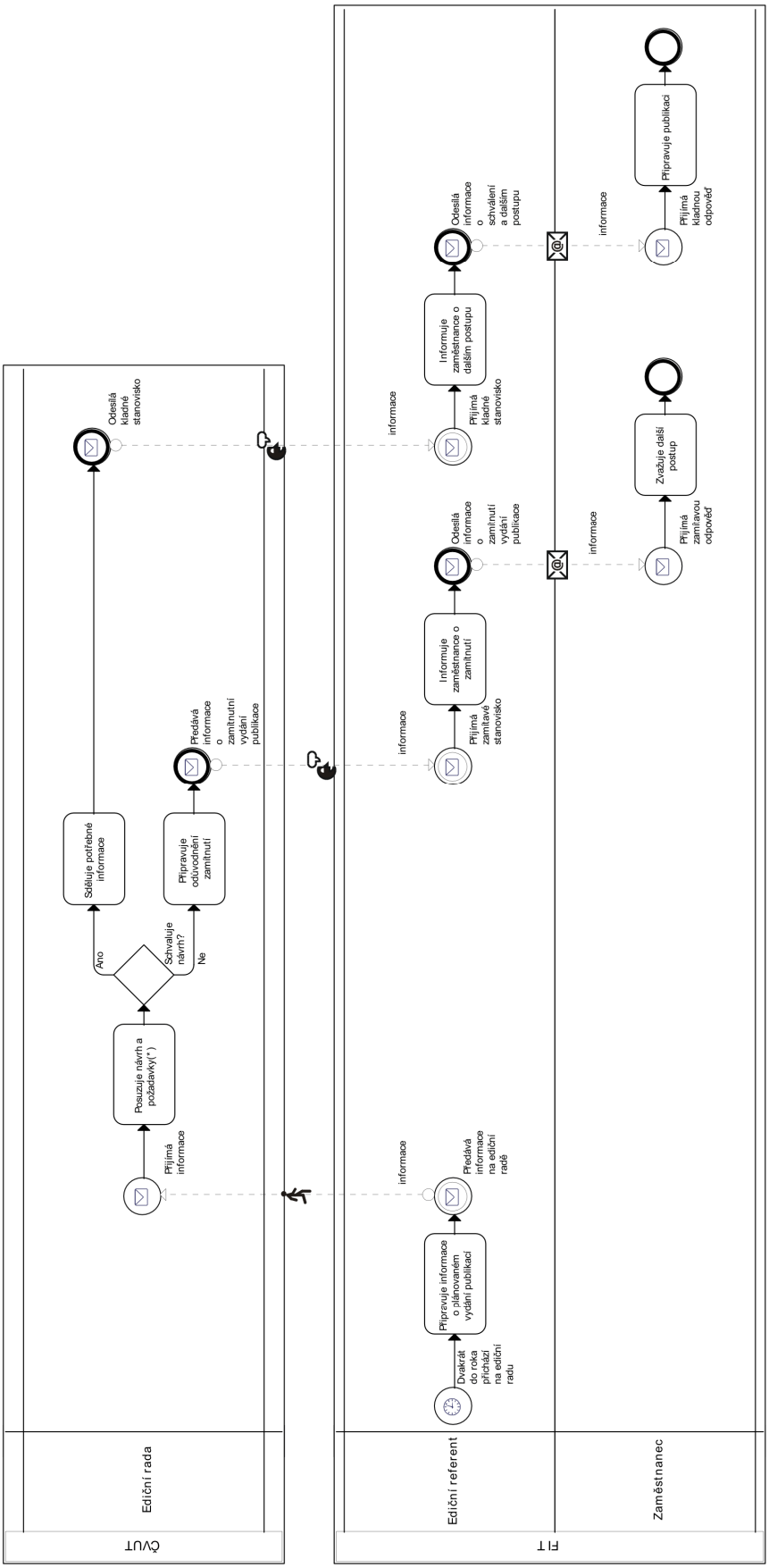
Pro každého identifikovaného participanta se v tomto kroku vytvoří modelová karta. Opět se jedná o strukturovaný text, jako v předchozím případě scénářů. Modelové karty zachycují konkrétního participanta kontextu ostatních participantů a všech scénářů, kterých se účastní včetně jeho rolí.

Collaborators:	Děkan	Zaměstnanec	Vedoucí katedry	Ediční rada ČVUT
Příprava vydání nové publikace (approves)	approves	initiates	approves	
Předložení žádosti o vydání publikace ediční radě ČVUT (is responsible)		is informed		approves

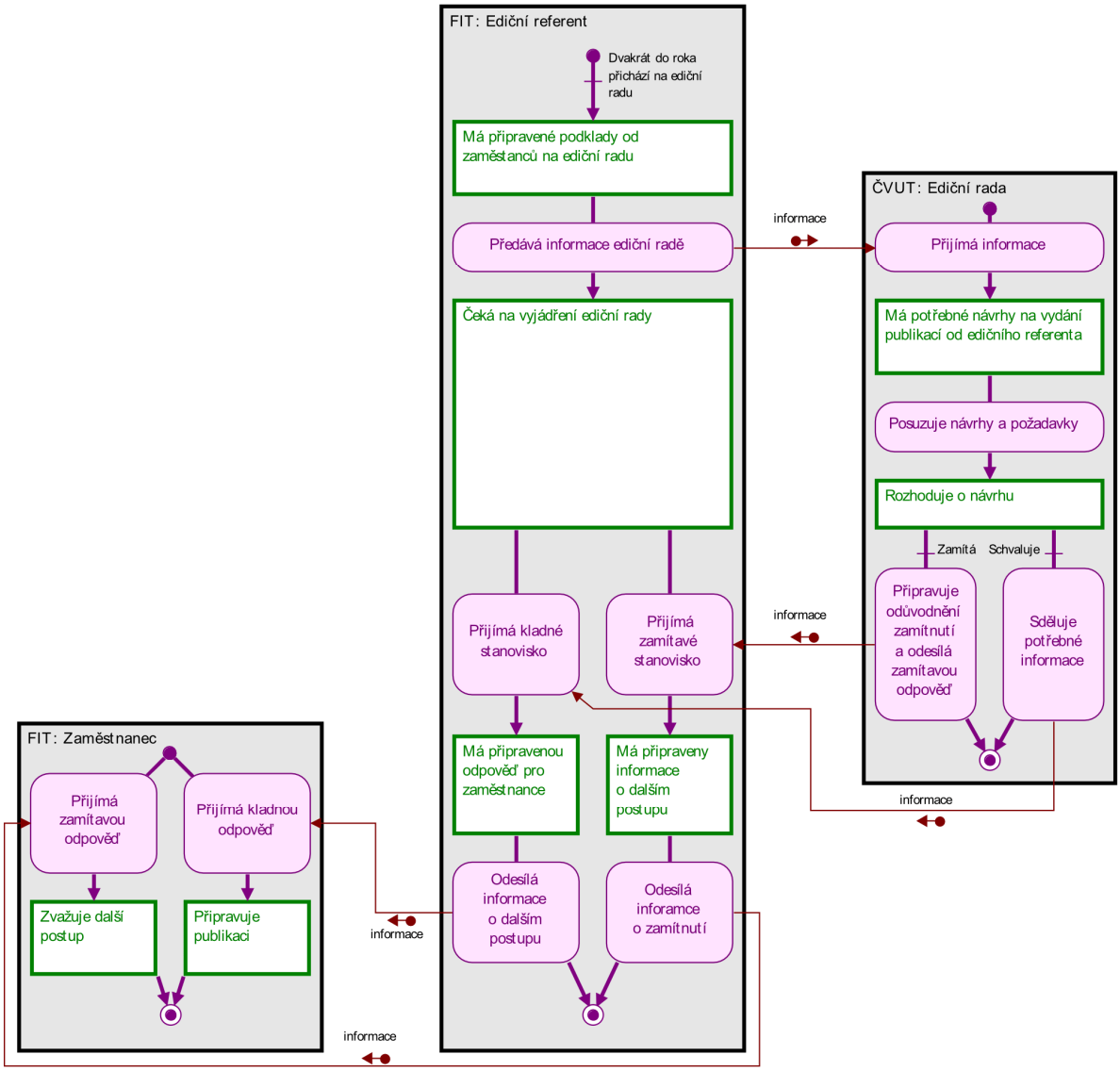
Obr. 11.4. Modelová karta participanta „Ediční referent“

Modelování procesů

Posledním krokem OBA, který není závislý na použitém nástroji pro modelování, je vytvoření procesních diagramů. Pro každý rozpoznaný proces a participanta se s použitím slovního popisu, scénáře a modelových karet sestaví životní cyklus objektu. Životní cyklus objektu je reprezentován procesním diagramem, který se skládá z aktivit, stavů a přechodů mezi těmito stavy, které náleží konkrétnímu participantovi.



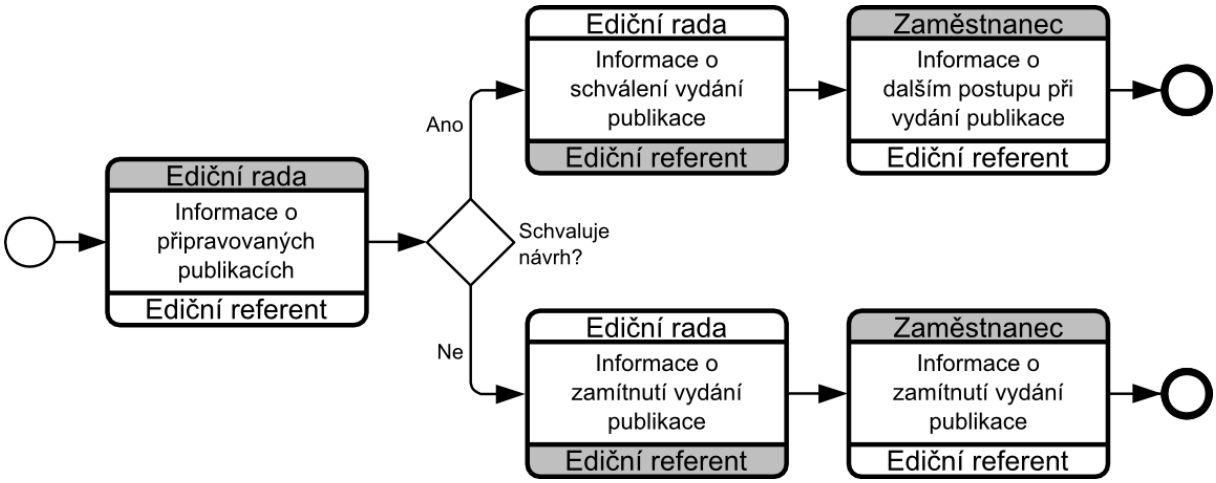
Obr. 11.5. Procesní diagram v BPMN pro proces „Předložení žádosti o vydání publikace ediční radě ČVUT“ – vlastní zpracování autora



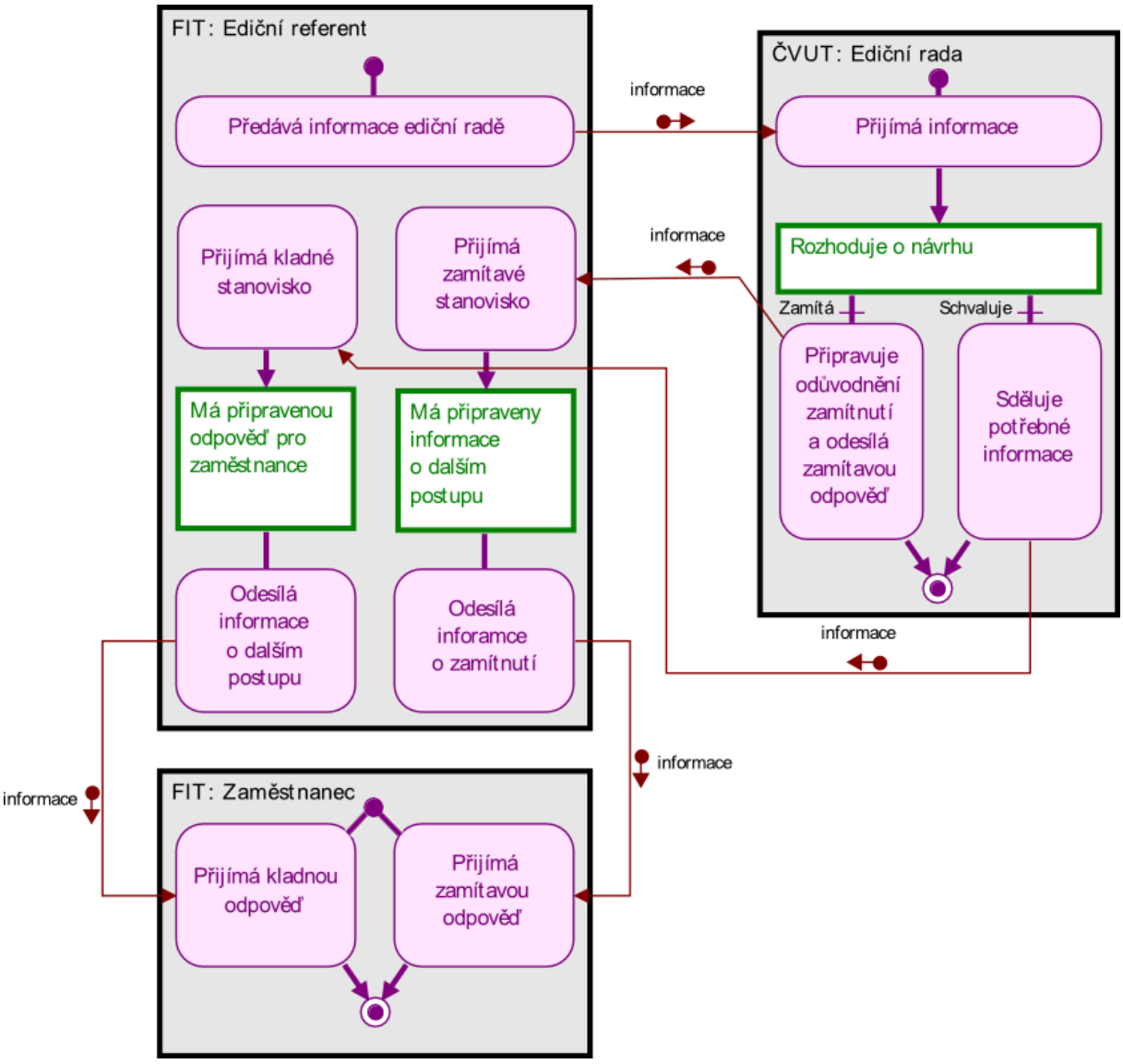
Obr. 11.6. Procesní diagram v BORM pro proces „Předložení žádosti o vydání publikace ediční radě ČVUT“ – vlastní zpracování autora

11.3.3 Model choreografie v BPMN a BORM

Procesní diagram v BORM je relativně jednoduchý a používá jen omezený počet pojmů a symbolů. Na rozdíl od BPMN neobsahuje BORM prvky, které by byly určeny pouze pro popis choreografie. Podle zvoleného modelovacího nástroje je možné vygenerovat tabulku komunikací, což je nejblíže reprezentaci, jak ji umožňuje BPMN.



Obr. 11.7. Diagram choreografie procesu „Předložení žádosti o vydání publikace ediční radě ČVUT“ v BPMN – vlastní zpracování autora



Obr. 11.8. Jedna z možností znázornění choreografie v BORM – vlastní zpracování autora

Záznam choreografie pomocí procesního diagramu, jak je prezentován na obrázku 11.8. není po formální stránce korektní. Jak již bylo dříve uvedeno, v procesním diagramu musí po každé aktivitě následovat stav a naopak, což zde není dodrženo. Další možností, jak zachytit přehled komunikací, je jednoduchá tabulka, kterou je možné na základě procesního diagramu vygenerovat. Struktura informací uvedených v tabulce 11.2. se blíží reprezentaci, jak ji umožňují diagramy choreografie v BPMN, nicméně je to stále nedostačující. Možným východiskem by bylo využít konceptu modelových karet z metody OBA a upravit jejich strukturu tak, aby umožňovala zapsat choreografii. Rozšíření metody OBA pro BORM je věnována následující kapitola.

Odesílatel	Příjemce	Zpráva	Podmínka
Ediční referent	Ediční rada	Informace o připravovaných publikacích	
Ediční rada	Ediční referent	Informace o schválení vydání publikace	Schvaluje návrh
Ediční referent	Zaměstnanec	Informace o dalším postupu při vydání publikace	Schvaluje návrh
Ediční rada	Ediční referent	Informace o zamítnutí vydání publikace	Neschvaluje návrh
Ediční referent	Zaměstnanec	Informace o zamítnutí vydání publikace	Neschvaluje návrh

Tabulka 11.2. Přehled komunikací mezi účastníky procesu v BORM

11.4 Návrh rozšíření metody OBA pro BORM

Jak bylo naznačeno v předchozí kapitole, autor této práce odhalil nedostatečnou vyjadřovací schopnost metody BORM v oblasti popisu komunikací participujících objektů v rámci procesu, respektive v možnostech záznamu choreografie procesu. Zároveň byla již v kapitole 10 prokázána značná orientace procesních diagramů v BORM na stavy. Jelikož současná podoba přístupu k rozpoznání participantů a definice procesů v podobě upravené metody OBA tato zjištění nereflektuje, navrhujeme její rozšíření o následující dva kroky.

11.4.1 Identifikace klíčových stavů objektů

Autoři původní metody OBA (Rubin a Goldberg, 1992) uvádějí, že je nutné v rámci posledního kroku metody pro každý objekt definovat množinu stavů, ve kterých se v průběhu životního cyklu objekt může nacházet. Životní cyklus objektu vyjadřuje změny jeho stavu v průběhu času. Stavy jednotlivým objektům se přidělují na základě scénářů. Oproti tomu, upravená metoda OBA jak je použita v BORM, definuje stavy objektů až v rámci tvorby procesních diagramů, což je podle názoru autora této práce méně efektivní, než kdyby k definici stavů objektů došlo ještě před tvorbou diagramů. Jak uvádějí (Shlaer a Mellor, 1992), (Taylor, 1995) a (Merunka, 2012) je důležité nejprve identifikovat stavy, ve kterých se jednotliví participanti nacházejí a až následně je možné identifikovat chování, respektive aktivity, které je nutné vykonat proto, aby participant přešel z aktuálního stavu do stavu nového.

Jak bylo dokázáno v kapitole 10, lze procesní digram BORM zapsat pomocí konečného automatu, který je kompozicí konečných automatů z nichž každý odpovídá právě jednomu participantovi procesu. Možnost popsat beze zbytku procesní diagramy pomocí konečného automatu poukazuje na silnou orientaci ORD diagramů na stavy. Autor této práce pro navrhuje rozšířit upravenou metodu OBA, kterou BORM využívá o krok, vycházející z posledního kroku původní metody jak ho uvádějí (Rubin a Goldberg, 1992).

Nově definovaný krok nazveme *Identifikace klíčových stavů objektů*. Jelikož metoda OBA využívá pro strukturovaný zápis textu upravených CRC karet, nebudeme zavádět jiný způsob zápisu. Pro popis klíčových stavů objektu participujícím na konkrétním procesu využijeme upravenou verzi „karet“, které se používají pro zápis scénářů. Výstupem nově zavedeného kroku OBA bude sada *Modelových karet stavů objektu* (Object State Modeling Card).

Při sestavování karet stavů objektů budeme vycházet verbálního popisu procesu, který byl získán pomocí interview a ze scénáře, který popisuje průběh procesu. Pro každého participanta tedy bude nutné definovat takový počet modelových karet stavů objektu, který odpovídá počtu procesů, kterých se daný participant účastní. Aby byla zaručena konzistence, navrhujeme svázat každou modelovou kartu stavů s konkrétním scénářem.

Každá *Modelová karta stavů objektů* bude obsahovat následující informace:

- *Název karty*
- *Název participanta* – bude obsahovat označení participanta, které musí být konzistentní v rámci všech následujících i předešlých kroků

- *Klíčové stavy objektu* – Na základě interview a patřičného scénáře budou identifikovány klíčové stavy participanta, které je nutné v rámci procesu zachytit. Na pořadí stavů záleží, protože vyplývá z životního cyklu objektu.
- *Scénář*, ke kterému karta náleží

Modelová karta stavů objektu - Ediční referent
participant: Ediční referent FIT
states: Má připravené podklady od zaměstananců na ediční radu Čeká na vyjádření ediční rady Má připravené informace pro zaměstnance o dalším postupu ve věci jeho žádost
Derived diagrams: Předložení žádosti o vydání publikace ediční radě ČVUT

Obr. 11.9. Modelová karta stavů objektu pro participanta „Ediční referent" – vlastní zpracování autora

Využijeme-li aparát, který jsme zavedli v kapitole 10, bude výstupem tohoto kroku úplná, konečná, neprázdná množina všech stavů $\mathbb{S}^i$ participanta $\mathbb{P}^i$.

11.4.2 Záznam komunikací pomocí OCMC

Rozmach distribuovaných technologií a potažmo servisně orientovaných architektur umožňuje snadnější interakce napříč společnostmi. Využití webových služeb může vést ke zjednodušení procesů a snížení provozních nákladů na proces alokovaných. Na druhé straně je při návrhu procesů obsahujících jak reálné uživatele (lidi) tak i webové služby nutné ošetřit různá chování webových služeb na konkrétní vstupy a pochopitelně také výjimky, aby nedocházelo k zacyklení nebo uváznutí procesu. Součástí Business Procees Managementu je řízení průběhu procesů, ať již jsou jejich účastníci reální uživatelé nebo webové služby. Problematikou užití webových služeb jako účastníků procesů se zabývá orchestrace a choreografie.

Procesní diagram v BORM používá, oproti jiným metodám a jazykům pro modelování procesů, jen omezený počet pojmů a symbolů. Například BPMN obsahuje prvek *Úloha v choreografii*, který je přímo určen pro popis choreografie. BORM takovým prvkem nedisponuje a přímo nezahrnuje možnost popsat komunikace obdobným způsobem, jak to umožňují diagramy choreografie v BPMN. Aby metoda BORM obstát v konkurenci jiných častěji používaných, metod a aby ji bylo možné jednoduše používat také v případech modelování

orchestrace a choreografie, jako je tomu například u BPMN, považuje autor této práce za vhodné a prospěšné definovat způsob zápisu orchestrace respektive choreografie.

Možným východiskem je využít konceptu modelových karet z metody OBA a upravit jejich strukturu tak, aby umožňovala zapsat choreografii. Tyto nové karty budeme označovat jako *Modelové karty choreografie objektu* (OCMC – Object Choreography Modeling Card). Pro popis vzájemných komunikací participantů v rámci procesu navrhujeme rozšířit metodu OBA pro BORM o krok – *Záznam komunikací pomocí OCMC*, který bude předcházet vytváření procesních diagramů.

Při sestavování modelových karet orchestrace objektu budeme opět vycházet verbálního popisu procesu, který byl získán pomocí interview a ze scénáře, který popisuje průběh procesu. Každá OCMC bude obsahovat matici komunikací, kde řádky budou vyjadřovat jednotlivé kroky procesu a sloupce budou obsahovat ostatní účastníky procesu. Pro každého participanta tedy bude definován takový počet modelových karet orchestrace objektu, který odpovídá počtu procesů, kterých se daný participant účastní. Aby byla zaručena konzistence, navrhujeme svázat každou modelovou kartu choreografie objektů s konkrétním scénářem.

Každá *Modelová karta choreografie objektu* bude obsahovat:

- *Název karty*
- *Matici komunikací*, kde každý řádek bude odpovídat jedné aktivitě daného participanta a sloupce, které budou obsahovat ostatní účastníky procesu. Jednotlivé buňky matice pak mohou obsahovat jeden ze symbolů (+,–) reprezentujících směr komunikace a případný popis předávaných dat. Symboly vyjadřující směr komunikace vycházejí z konvence, kterou jsme zavedli v kapitole 10 pro vysílané a přijímané zprávy. Symbol „+“ bude reprezentovat přijímané zprávy a symbol „–“ zprávy vysílané.
- *Scénář*, ke kterému karta náleží

Modelová karta choreografie objektu - Ediční referent			
participant: Ediční referent FIT			
communications:			
		Ediční rada ČVUT	Zaměstnanec
Předává informace ediční radě		„–“ Informace	
OR	Přijímá kladné stanovisko	„+“ Informace	
	Odesílá informace o dalším postupu		„–“ Informace
	Přijímá zamítavé stanovisko	„+“ Informace	
	Odesílá informace o zamítnutí		„–“ Informace
Derived diagrams: Předložení žádosti o vydání publikace ediční radě ČVUT			

Obr. 11.10. Modelová karta choreografie objektu pro participanta „Ediční referent“ – vlastní zpracování autora

Každá *Modelová karta choreografie objektu* tedy bude reprezentovat množinu všech přijímaných a vysílaných zpráv $\mathbb{M}^i = -\mathbb{M}^i \cup +\mathbb{M}^i$ participanta $\mathbb{P}^i$.

Agregací modelových karet choreografie objektu náležících jednotlivým participantům $\mathbb{P}^i$ konkrétního procesu, získáme model choreografie takového procesu, kde $\Phi = \bigcup_i -\mathbb{M}^i, \Omega = \bigcup_i +\mathbb{M}^i$. Obdobným způsobem lze získat model choreografie celé problémové domény.

Modelová karta choreografie procesu - Předložení žádosti o vydání publikace ediční radě ČVUT				
participant: Ediční referent FIT, Ediční rada ČVUT, Zaměstnanec				
communications:				
		Ediční referent FIT	Ediční rada ČVUT	Zaměstnanec
Předává informace ediční radě		„–“ Informace	„+“ Informace	
Přijímá informace		„+“ Informace	„–“ Informace	
OR	Sděluje potřebné informace	„+“ Informace	„–“ Informace	
	Přijímá kladné stanovisko	„+“ Informace	„–“ Informace	
	Odesílá informace o dalším postupu	„–“ Informace		„+“ Informace
	Přijímá kladnou odpověď	„–“ Informace		„+“ Informace
	Připravuje odůvodnění zamítnutí a odesílá odpověď	„+“ Informace	„–“ Informace	
	Přijímá zamítavé stanovisko	„+“ Informace	„–“ Informace	
	Odesílá informace o zamítnutí	„–“ Informace		„+“ Informace
	Přijímá zamítavou odpověď	„–“ Informace		„+“ Informace
Derived diagrams: Předložení žádosti o vydání publikace ediční radě ČVUT				

Obr. 11.11. Agregovaná Modelová karta choreografie objektu procesu „Předložení žádosti o vydání publikace ediční radě ČVUT“ – vlastní zpracování autora

Reprezentace vzájemných komunikací účastníků procesu pomocí matice je pro potřeby následné implementace dostačující. Nicméně předmětem další diskuze by mělo být, zda by nebylo vhodné rozšířit sadu symbolů metody BORM o nový prvek, určený pro zobrazení komunikací v rámci choreografie podobně jak je tomu v případě BPMN.

11.4.3 Rozšířená metoda OBA

Původních pět kroků metody OBA jak je uvádí (Merunka, 2008) zůstane zachováno. Metodu rozšíříme o dva kroky zaměřené na identifikaci klíčových stavů jednotlivých objektů a na podrobný popis komunikací mezi participujícími

objekty procesů. Rozšířená metoda OBA pro použití v BORM, jak ji navrhuje autor této práce pak bude mít následujících sedm kroků:

- **Rozpoznání procesů.** V tomto kroku 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 vymezit zadání v rámci možného širšího kontextu řešeného problému.
- **Rozpoznání plánování scénářů jako detailního popisu 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.
- **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.
- **Identifikace klíčových stavů.** Pro každý rozpoznaný objekt ze druhého kroku metody se vytvoří jeho *Modelová karta* stavů objektu, která obsahuje jméno objektu, seznam klíčových stavů objektu v souvislosti s konkrétním scénářem a příslušnost k danému scénáři. Na pořadí stavů záleží, protože vyplývá z životního cyklu objektu.
- **Definování komunikací mezi objekty v rámci jednotlivých procesů pomocí OCMC.** V tomto kroku se pro každý rozpoznaný objekt ze druhého kroku vytvoří jeho *Modelová karta choreografie objektu*, která obsahuje jméno objektu a matici komunikací objektu s ostatními participanty procesu. Řádky matice odpovídají jednotlivým krokům procesu. Sloupce pak představují ostatní participanty. Jednotlivé buňky matice pak mohou obsahovat jeden ze symbolů reprezentujících typ a směr komunikace. Abychom zaručili konzistenci celé analýzy, obsahuje každá OCMC také referenci na konkrétní scénář.
- **Modelování procesů.** Zde 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.

- **Verifikace a validace.** Zde se kontroluje shoda mezi diagramy, tabulkami a skutečnými požadavky na systém. K tomu slouží dva nástroje. Jedním z nich je datový model obsahující skutečná data, pomocí nichž lze prověřit správnost návrhu. Druhým nástrojem je simulátor procesů (platí pro nástroj CraftCASE), který dovoluje procházet jednotlivé kroky procesu znázorněného diagramy a umožňuje tak odhalit zacyklení nebo případná uváznutí procesu.

Autor této práce navrhl rozšíření metody OBA pro BORM o dva kroky, které zpřesňují návrh procesů na základě identifikace klíčových stavů objektů a zobrazení komunikací pomocí OCMC. Rozšíření o krok *Identificae klíčových stavů* reflektuje povahu procesních diagramů v BORM, tedy jejich orientaci na stavy. Druhé rozšíření v podobě zavedení *Modelových karet choreografie objektu* reflektuje potřebu mít k dispozici prostředek pro popis choreografie, jak je tomu například v BPMN. Navrhovaná rozšíření spadají do kategorie aktivit předcházejících návrhu procesních diagramů (tzv. pre-modeling activities). Proto je nutné provést oba kroky ještě před samotným návrhem diagramů, což dovoluje odhalit případné nedostatky a nekonzistence, které přímo nevyplývaly z verbálního popisu problémové domény.

Z hlediska typologie uživatelů procesních diagramů jsou praktické dopady navrhovaných rozšíření následující:

- *Vlastník procesu* – je osoba, která za daný proces odpovídá. Vlastník je odpovědný za udržitelnost procesu, tedy za jeho dlouhodobé a efektivní fungování. Jeho úkolem je také průběžná aktualizace a zlepšování procesu. Jelikož přichází do styku s již hotovým procesním diagramem, nemají navrhovaná rozšíření přímý vliv na vykonávání jeho funkce. Vlastník procesu je též někdy označován jako správce/administrátor.
- *Účastník procesu* – je subjekt, který se procesu účastní a přímo se podílí na jeho vykonávání. Účastníky můžeme rozdělit na *externí* a *interní*. *Externími účastníky* jsou takové subjekty, které nejsou součástí organizace, které se proces týká. Analogicky pak *interní účastníci* budou subjekty, které jsou součástí dané organizace. Subjektem může být konkrétní osoba, skupina osob, organizace, webová služba, informační systémy atp. Obdobně jako v případě *vlastníka procesu* nemají navrhovaná rozšíření přímý vliv ani na účastníky procesu.
- *Analytik/IT architekt* – je osoba, která na základě interview s vlastníky procesů a účastníky procesů sestavuje seznam požadovaných funkcí systému. Získané podklady dále strukturuje s použitím metody OBA.

Jelikož se navrhovaná rozšíření přímo týkají metody OBA a spadají do kategorie aktivit, které nezbytné vykonat ještě před návrhem procesních diagramů, přímo ovlivňují práci analytiků/IT architektů. Zavedením *modelových karet stavů objektu* reflektuje autor této práce potřebu nejprve identifikovat stavy (Shlaer a Mellor, 1992), (Taylor, 1995) a (Merunka, 2012), ve kterých se jednotliví participanti nacházejí. Na základě stavů lze odvodit aktivity, které je nutné vykonat proto, aby participant přešel z aktuálního stavu do stavu nového. Identifikace stavů ještě před vytvářením diagramů poskytuje kontrolu správnosti pochopení verbálního popisu problému a umožňuje lépe se zaměřit na požadovaný výsledek procesu. Záznam komunikací mezi participanty v rámci jednotlivých procesů pomocí *modelových karet choreografie objektu* (OCMC) rozšiřuje možnosti metody BORM v zápisu choreografie, které by bez zavedení OCMC byly velmi omezené.

- *Vývojář/programátor* – je osoba, která bude na základě zpracované procesní analýzy, datového modelu a dalších podkladů od analytiků a IT architektů implementovat požadovanou funkčnost. Metoda BORM disponuje nástroji pro procesní a datové modelování. Popis komunikací čistě pomocí procesních diagramů se v praxi ukázaly ne vždy dostačujícím prostředkem. Autor této práce se podílej jak přímo tak také jako IT analytik na implementaci systémů, kde jako podklad pro práci sloužily především procesní diagramy. Mnohdy dochází ke špatné interpretaci procesních diagramů ze stany vývojářů. Autor této práce se proto domnívá, že by měli být co nejpřesněji definovány nejen funkce, které má systém plnit, ale také komunikace. Nejen v případech, kdy se procesů účastní webové služby je důležité co nejpřesněji definovat komunikace a datové toky. Jako vhodné řešení se jeví strukturovat popis komunikací pomocí matice komunikací, jak je tomu v případě navrhovaného zavedení OCMC. Ze zkušeností autora této práce z praxe je taková reprezentace komunikací pro vývojáře lépe uchopitelná. Specifickým případem je využití webových služeb jako *účastníků procesu*, kdy je nezbytné co nejpřesněji definovat jak pořadí komunikací tak i jejich obsah.
- *Systémový integrátor* – je osoba nebo firma, jejímž úkolem je provázat a propojit jednotlivé heterogenní komponenty informačního systému do jednoho homogenního celku tak, aby odpovídal potřebám uživatele a podpořil jeho procesy. Jelikož stojí systémový integrátor na pomezí architekta a realizátora informačního systému, platí pro něho vše co bylo uvedeno výše. Systémový integrátor by měl zastat jak roli architekta

a navrhnout podobu budoucího informačního systému tak i roli vývojáře, který tento svůj návrh realizuje buď s použití již hotových komponent a stávajících částí informačního systému, a nebo vytvořením nových komponent.

Výše uvedená typologie uživatelů procesních diagramů ukazuje, že cílovými skupinami uživatelů, kterým navrhovaná rozšíření přinesou největší užitek budou analytici/IT architekti a programátoři/vývojáři. Úkolem systémového integrátora je propojit komponenty informačního systému podniku do homogenního celku. Jednotlivé komponenty spolu musí komunikovat, a proto navrhovaná rozšíření, zejména v oblasti přesnějšího definování komunikací jednotlivých účastníků procesu, povedou ke zjednodušení a zefektivnění práce systémových integrátorů. Zbývající dvě skupiny uživatelů mohou těžit z výhod navrhovaných rozšíření pouze v přídě že analytici/IT architekti a také programátoři/vývojáři odvedou svoji práci dobře.

12 Diskuse

V posledních letech došlo k výraznému zvýšení poptávky po metodách a nástrojích pro BPM a také obecně ke zvýšení zájmu o oblast řízení podnikových procesů. Rozvoj síťových a internetových technologií umožňuje jednoduše zpřístupnit část vlastního systému obchodním partnerům pomocí webových služeb. Podle vlastních zkušeností autora se při návrhu informačních systémů nebo softwaru obecně stále více uplatňuje přístup, kdy se vytváří software na základě analýzy interních obchodních procesů organizace a ne, jak tomu bývalo, že se obchodní procesy přizpůsobovaly používanému software. Právě komplexní analýza procesů je předpokladem pro vytvoření plnohodnotného softwaru, který bude efektivně podporovat obchodní procesy ve firmě. Jedním z účastníků procesu mohou být také webové služby, což může vést ke zjednodušení návrhu procesu a jeho vyšší efektivitě.

Autoři zabývající se metodou BORM (Knott, a další, 2006) a (Merunka, 2008), často zmiňují, že procesní diagram v BORM je tvořen objekty (participanty), kde každý participant je modelován jako Mealyho konečný automat. Nicméně, zatím neexistoval formální zápis procesních diagramů pomocí konečného automatu. Jak bylo dokázáno v kapitole 10, lze procesní diagram v BORM popsat pomocí konečného automatu, jehož konstrukce byla odvozena na základě vybraných modelů konečných automatů popsaných v kapitole 5. Otevírá se tím možnost využití 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.

Jako prostor pro další práci se nabízí možnost dalšího rozšíření metody BORM o asynchronní komunikace. Asynchronní komunikace nevyžadují přijetí zprávy ve stejné chvíli kdy je odesílána a pokud bude velikost kanálu pro předávání zpráv provázána s konkrétním stavem, lze stále výsledný model simulovat pomocí konečného automatu. Návrh takového rozšířený je možný, protože jsou dobře zdokumentovány postupy vzájemných transformací různých druhů konečných automatů mezi sebou (Peng a Puroshothaman, 1991).

Metoda BORM není ani zdaleka tak široce používána jako, v této práci často zmiňovaná, notace BPMN. BORM zatím neobsahuje konkrétní nástroje a přístup jak choreografie a orchestrace modelovat. Jistě, je možné popsat komunikace mezi webovými službami účastnící se procesů, ale způsob pro záznam takových

komunikací zatím definován nebyl. Proto autor této práce navrhl rozšíření metody OBA, kterou BORM využívá při definici procesů, tak aby bylo možné choreografii a orchestraci lépe a efektivněji zaznamenat. Konkrétní návrh rozšíření je prezentován v kapitole 11.4. Prostor pro další práci je zejména v oblasti implementace navrhovaného rozšíření metody OBA do některého z nástrojů, které využívají metodu BORM. Ideálním kandidátem se jeví IZMAN Case, který byl vyvinut na katedře informačního inženýrství, jejímž členem je autor této práce.

13 Závěr

Práce byla změřena na problematiku řízení komunikace mezi interními a externími účastníky obchodních procesů s využitím metody BORM. Jednotliví participanti mohou být reprezentováni jak webovými službami, tak i reálnými uživateli. Obecně je takový způsob řízení komunikace v rámci procesů nazýván orchestrací respektive choreografií. Autor se tedy zabýval možnostmi rozšířit metodu BORM, tak aby lépe vyhovovala nároku kladeným na moderní metody popisu procesů.

Hlavním cílem této práce bylo vyplnit mezeru v teoretických základech metody BORM, která spočívala v absenci jednoznačného formálního popisu procesních diagramů. Autoři (Knott, a další, 2003) a (Merunka, 2008), kteří se metodě BORM intenzivně věnují shodně uvádějí, že jsou procesní diagramy založeny na fundamentálních principech konečných automatů. Žádný z výše uvedených autorů však neuvádí a ani neodkazuje na konkrétní popis procesních diagramů s využitím konečných automatů. Autor této práce nalezl a použil analogické koncepty z oblasti konečných automatů k sémantice procesních diagramů v BORM tak, aby bylo možné definovat jejich formální popis a bylo možné tímto způsobem zaznamenat nejen řídící toky, ale také datové toky.

Aby mohlo být tohoto cíle dosaženo bylo nutné splnit následující dílčí cíle:

- Analyzovat literární zdroje zabývající se problematikou procesního modelování, teorie konečných automatů a webových služeb. Byly popsány vybrané nástroje a techniky z výše jmenovaných oblastí.
- Na základě analýzy publikací z oblasti teorie konečných automatů, nalezení analogického konceptu komunikací k sémantice komunikací v BORM a následné syntézy tohoto konceptu s modelem Mealyho konečného automatu, byl vytvořen formální popis procesního diagramu v BORM (kapitola 10).
- Pomocí tohoto formálního aparátu, bylo následně možné odvodit popis modelu orchestrace a choreografie v BORM (kapitola 11.1). Porovnáním obou modelů, dospěl autor této práce k závěru, že v případě metody BORM lze pohlížet na orchestraci jako na zvláštní případ choreografie.

Byla zpracována případová studie (kapitola 11.3), která na příkladu z praxe srovnává možnosti BORM a BPMN v oblasti modelování orchestrace a choreografie. Dále je zde zdůvodněno proč byla pro porovnání zvolena notace BPMN. Případová studie odhalila nedostatky metody BORM v oblasti zachycení choreografie a pomohla určit konkrétní oblast, která byla následně předmětem rozšíření.

Dílčím cílem práce bylo rozšířit metodu BORM případně její části tak, aby bylo možné lépe a efektivněji modelovat choreografii procesů. V rámci naplnění toho cíle byly navrženy následující rozšíření metody BORM:

- Na základě komparace původní podoby metody OBA a upravené verze, kterou využívá BORM spolu s prokázáním silné orientace procesních diagramů na stavy bylo navrženo rozšíření metody OBA o krok, který se zabývá identifikací klíčových stavů objektů (podrobně popsáno v kapitole 11.4.1).
- Aby bylo možné efektivněji zachytit choreografii v rámci procesů, bylo navrženo další rozšíření metody OBA o krok ve kterém jsou vytvářeny tzv. *Modelové karty choreografie objektu*, které pomocí maticového zápisu zachycují komunikace konkrétního participanta s ostatními účastníky procesu (podrobně popsáno v kapitole 11.4.2).

Navrhovaná rozšíření byla aplikována na konkrétní příklad z praxe (kapitola 11.4) a navazují na případovou studii, která byla zpracovaná v kapitole 11.3. Rozšíření metody OBA spadají do kategorie aktivit předcházejících návrhu procesních diagramů (tzv. pre-modeling activities). Proto je nutné provést oba výše zmiňované kroky ještě před samotným návrhem diagramů, což dovoluje odhalit případné nedostatky a nekonzistence, které přímo nevyplývaly z verbálního popisu problémové domény. Z hlediska typologie uživatelů procesních diagramů mají navrhovaná rozšíření největší přínos pro analytiky, kteří procesní diagramy sestavují a pro programátory, kteří následně procesy implementují. Navrhovaná rozšíření, zejména v oblasti přesnějšího definování komunikací jednotlivých účastníků procesu, povedou ke zjednodušení a zefektivnění práce systémových integrátorů. (podrobněji popsáno v kapitole 11.4.3).

Posledním dílčím cílem práce bylo na základě komparace vybraných jazyků pro popis webových služeb a jazyku, který je použit pro definici procesů v nástroji IZMAN Case navrhnout změny v jazyce (formátu dat), který je aktuálně pro záznam procesů používán s ohledem na možnost definovat orchestraci a choreografii. Podrobně jsou výsledky popsány v kapitole 7.9. Pro splnění tohoto dílčího cíle bylo nutné:

- Srovnat vybrané jazyky, pro popis orchestrace a choreografie a nalézt mezi nimi takový koncept, který by se nejvíce přibližoval doposud používanému přístupu v nástroji IZMAN Case.
- Na základě srovnání jazyků byl zvolen jazyk BPEL, který se nejvíce přibližuje konceptu metody BORM a bylo navrženo rozšíření Ecore modelu metody BORM, který používá IZMAN Case pro implementaci metody BORM, o vybrané konstrukty jazyka BPEL, tak aby umožňoval lépe definovat komunikace mezi participanty. Následkem toho byl rozšířen jazyk založený na XML, pomocí kterého IZMAN Case ukládá informace o procesech.

Jednoznačný a formální popis procesního diagramu v BORM otevírá možnosti návazného výzkumu a dalších rozšíření metody BORM. Jedná se především o následující oblasti:

- Formální popis otevírá možnosti lepší implementace metody BORM v CASE nástrojích, především v konstrukci simulátoru procesů. Posledním krokem metody OBA je *Verifikace a validace*, v rámci něhož dochází k ověření formální správnosti navržených procesů. Simulátor procesů, který je k dispozici v nástroji CraftCASE bohužel není volně k dispozici a v případě nástroje IZMAN Case takový prostředek pro validaci a verifikaci procesu zcela chybí. Pro oblast konečných automatů byly podrobně popsány postupy jak ověřit, zda jsou všechny stavy účastníků dosažitelné a zda jsou všechny aktivity prováděny. Možnosti ověření správnosti návrhu konečného automatu zmiňují například (Gill, 1962), (Gouda a Yu, 1984) a (Wright, 2005). Dále je možné automatizovaně identifikovat stavy, ve kterých by mohlo dojít k uváznutí procesu a tím lze ověřit konzistenci modelu.
- Jelikož jsou dobře zdokumentovány postupy vzájemných transformací různých druhů konečných automatů mezi sebou, bylo by možné rozšířit metodu BORM o asynchronní komunikace, jak je popisována v souvislosti s komunikujícími konečnými automaty (Peng a Puroshothaman, 1991), jejichž prvky byly v této práci použity pro popis procesních modelů.

Asynchronní komunikace nevyžadují přijetí zprávy ve stejné chvíli kdy je odesílána a pokud bude velikost komunikačního kanálu provázána s konkrétním stavem, lze stále výsledný model simulovat pomocí konečného automatu. Cílem této práce nebylo rozšíření metody BORM o asynchronní komunikace, a proto se její autor touto problematikou nezabýval.

- Reprezentace vzájemných komunikací účastníků procesu pomocí maticového zápisu (podrobně popsáno v kapitole 11.4.2) je pro potřeby následné implementace dostačující. Předmětem další diskuze by mělo být, zda by nebylo vhodné rozšířit sadu symbolů metody BORM o nový prvek, který by byl určen pro zobrazení komunikací v rámci choreografie. Dle názoru autora této práce by bylo vhodné rozšířit sadu symbolů metody BORM, obdobným způsobem jak byla rozšířena notace BPMN o grafický prvek *Úloha v choreografii*.
- Navrženým rozšířením možností nástroje IZMAN Case byla nastolena otázka, zda by nebylo vhodné navrhnout postup mapování XML výstupu tohoto CASE nástroje na jazyk BPEL, který je v oblasti orchestrace procesů široce používán.

Citovaná literatura

ADLER, Mike. An algebra for data flow diagram process decomposition. *Software Engineering, IEEE Transactions on*, 1988, 14.2: 169-183.

ALONSO, Gustavo, a další. *Web services: concepts, architectures, and applications*. New York: Springer, 2004, 354 s. ISBN 35-404-4008-9.

BARKER, Adam, WALTON, Christopher, D. a ROBERTSON, David. Choreographing web services. *IEEE Transactions on Services Computing*, vol. 2, no. 2, s. 152-166, 2009

BARROS, Alistair, a další. Service Interactions Patterns. *In Proceedings of the 3rd International Conference on Business Process Management (BPM)*, Nancy, France, September 2005. Springer Verlag, s. 302-218

BARROS, Alistair, MARLON, Dumas a PHILLIPA Oaks. A Critical Overview of the Web Service Choreography Description Language (WS-CDL). *BPTrends Newsletter 3 (2005)*. [Online] 2005. [Citace: 2012-05-11]. Dostupné z: http://news.bptrends.com/publicationfiles/03-05%20WP%20WS-CDL%20Barros%20et%20al.pdf

BARROS, Alistair, MARLON, Dumas a PHILLIPA Oaks. Standards for Web Service Choreography and Orchestration: Status and Perspectives. *Business process management workshops: BPM 2005 international workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS*, Nancy, France, September 5, 2005 : revised selected papers. New York: Springer, 2006, s. 61-74. ISBN 978-3-540-32595-6.

BRAND, Daniel a ZAFIROPULO, Pitro. On Communicating Finite-State Machines. *Journal of the Association for Computing Machinery*, vol. 30, no. 2, s. 323-342, April 1983

BROCKE, Jan vom a ROSEMANN, Michael. *Handbook on business process management 1*. New York: Springer, 2010, p. cm. ISBN 978-364-2004-155.

BUSI, N., GORRIERI, R., GUIDI, C., LUCCHI, R. a ZAVATTARO, G. Choreography and orchestration conformance for system design. *In COORDINATION*, LNCS 4038, s. 63–81, 2006.

BUSI, N., GORRIERI, R., GUIDI, C., LUCCHI, R. a ZAVATTARO, G. Choreography and orchestration: a synergic approach for system design. *In ICSOC*, LNCS 3826, s. 228–240, 2005

BUSSLER, C. Enterprise Application Integration and Business-to-Business Integration Processes. In: *Process Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley Publishing (2005) s. 61-82

CERAMI, Ethan. *Web services essentials*. 1st ed. Sebastopol, CA: O'Reilly, 2002, xiii, 288 s. ISBN 05-960-0224-6.

COVER, Robin. *Web Services Conversation Language (WSCL)*. [Online]. 2002 [cit. 2012-02-08]. Dostupné z: http://xml.coverpages.org/wscl.html

DAVIS, Martin D, SIGAL Ron a WEYUKER Elaine J. *Computability complexity,and languages: fundamentals of theoretical computer science*. 2nd ed. San Francisco: Morgan Kaufmann Publishers, 1994, xvii, 609 s. ISBN 01-220-6382-1.

ESTUBLIER, J. a SANLAVILLE, S. Business Processes and Workflow Coordination of Web Services. *In IEEE International Conference on e-Technology, e-Commerceand e-Service*. 2005. Hong Kong

FENSEL, D. a BUSSLER, C. *The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications*. [Online] 2002. [Citace: 2012-07-01]. Dostupné z: http://www.wsmo.org/papers/publications/wsmf.paper.pdf

FREDLUND, L. Implementing WS-CDL. *In Proceedings of the second Spanish Workshop Web Technologies (JSWEB '06)*, 2006.

FU, X., BULTAN, T. a SU., J. Analysis of Interacting BPEL Web Services. *In Proceedings of the International Conference on the World Wide Web Conference (WWW)*, s. 621–630, New York, NY, USA, 2004. ACM Press.

GIAGLIS, G., M. A taxonomy of business process modeling and information systems modeling techniques. *Proceedings in International journal of Flexible Manufacturing Systems*, vol. 13, no. 2, s. 209-228, 2001.

GIRAULT, Claude a VALK Rüdiger. *Petri nets for systems engineering: a guide to modeling, verification, and applications*. New York: Springer, 2003, xvi, 607 p. ISBN 35-404-1217-4.

GILL, Arthur. *Introduction to the theory of finite-state machines*. 1. vyd. New York: McGraw-Hill, 1962, 207 s. ISBN 0070232431.

GOUDA, Mohamed a YAO-TIN, Yu. Synthesis of communicating finite-state machines with guaranteed progress. *Communications, IEEE Transactions on 32.7 (1984)*: s. 779-788.

HABÁŇ, Jaromír a SODOMKA, Petr. Jak rozumí systémové integraci dodavatelé informačních systémů?. *Systems integration 2004: 12th international conference,*

Prague, Czech Republic, June 14-15, 2004 : proceedings. Ed. 1. Editor Jan Pour. Praha: Vysoká škola ekonomická, 2004, 584 s. ISBN 80-245-0701-3.

HOLZMANN, Gerard J. *Design and validation computer protocols*. Vyd. 1. New Jersey: Prentice-Hall, 1991, 500 s. ISBN 01-353-9925-4.

HOPCROFT, John E. *Introduction to automata theory, languages, and computation*. 2nd ed. Boston: Addison-Wesley, 2001, xiv, 521 s. ISBN 0-201-44124-1.

HUFFMAN, David A. The synthesis of sequential switching circuits. *Journal of the Franklin Institute*, 1954, 257.3: 161-190.

CHANDY, K., M. Event-driven applications: Costs, benefits and design approaches. *Gartner Application Integration and Web Services Summit 2006*, San Diego, CA, June 2006.

JENSEN, Kurt. Coloured petri nets. In: *Petri nets: central models and their properties*. Springer Berlin Heidelberg, 1987. p. 248-299.

JESTON, John a NELIS, Johan. *Business process management: practical guidelines to successful implementation*. 2nd ed. Amsterdam: Elsevier, 2008, xxvii, 469 s. ISBN 978-0-75-068656-3.

JING, Li, JIFENG, He, HUIBIAO, Zhu a GEGUANG, Pu. Modeling and verifying web services choreography using process algebra*. In SEW '07: Proceedings of the 31st IEEE Software Engineering Workshop*, s. 256–268, Washington, DC, USA, 2007. IEEE Computer Society.

KNOTT, Roger, MERUNKA, Vojtech, POLÁK, Jiri. The BORM methodology: a third-generation fully object-oriented methodology. *Knowledge-Based Systems*, 2003, 16.2: 77-89.

KNOTT, Roger, P., MERUNKA, V., POLÁK, J. Process Modeling for Object Oriented Analysis using BORM Object Behavioral Analysis. *In Proceedings of ICRE2000 – Fourth International Conference on Requirements Engineering*, Chicago 2000

LIU, Liping a ROUSSEV Boris. *Management of the object-oriented development process*. Hershey: Idea Group Pub., 2006, s. 345-366. ISBN 1-59140-604-8.

MANDRIOLI, Dino a GHEZZI Carlo. *Theoretical foundations of computer science*. New York: Wiley, 1987. ISBN 04-718-3834-9.

MARTENS, A. AnalyzingWeb Service Based Business Processes. *In Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering (FASE)*, s. 19–33, Barcelona, Spain, April 2004, Springer-Verlag.

MEALY, George H. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 1955, 34.5: 1045-1079.

MENDLING, J. a HAFNER, M. From WS-CDL Choreography to BPEL Process Orchestration. *In Journal of Enterprise Information Management (JEIM)*. Special Issue on MIOS Best Papers, 2008.

MERUNKA, Vojtěch. FSM-Based Object-Oriented Organization Modeling and Simulation. *Lecture Notes in Business Information Processing*, Advanced Information Systems Engineering Workshops, 2012, s. 398-412. ISBN 978-3-642-31068-3

MERUNKA, Vojtěch. *Objektové modelování*. 1. vyd. Praha: Alfa Nakladatelství, 2008, 197 s. ISBN 978-80-87197-04-2.

MOORE, Edward F. Gedanken-experiments on sequential machines. *Automata studies*, 1956, 34: 129-153.

MORAVEC, Josef a BROŽEK, Jiří. Experience with business process orchestration and choreography using BORM methodology. *Systémová integrace*, 2012, roč. 19, č. 3, s. 63-72. ISSN: 1210-9479.

MORAVEC, Josef a PAPÍK, Martin. Transformace BORM-Petriho síť s využitím supervize. *Objekty 2010 : sborník příspěvků jubilejního 15. ročníku konference*. 1. vyd. Ostrava : Ostravská univerzita, 2010. s. 335. ISBN 978-80-7368-899-8.

MURATA, T. Petri nets: Properties, analysis and applications. *In Proceedings of the IEEE*, pages 541–580, 1989.

OASIS WSBPEL. *Web Services Business Process Execution Language Version 2.0.* [Online] 2007. [Citace: 2012-04-17]. Dostupné z: http://docs.oasis-open.org/wsbpel/2.0/
wsbpel-v2.0.html

OASIS WS-HumanTask. *Web Services – Human Task Specification Version 1.1*. [Online] 2010. [Citace: 2012-05-20]. Dostupné z: http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-spec-cs-01.html

OMG BPMN. *BPMN 2.0. BPMN Specification*. [Online] 2011. [Citace: 2012-07-01]. Dostupné z: http://www.omg.org/spec/BPMN/2.0/

OMG MOF. *MetaObject Facility*. [Online] 2012. [Citace: 2012-04-08]. Dostupné z: http://www.omg.org/mof/

OULD, Martyn A. *Business process management: a rigorous approach*. Swindon: British Computer Society, 2005, xiv, 346 s. ISBN 9780929652276.

PELTZ, C. Web Services Orestrestration and Choreography. *IEEE Computer*, vol. 36, no. 10, s. 46-52, October 2003.

PENG, Wuxu; PUROSHOTHAMAN, S. Data flow analysis of communicating finite state machines. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1991, 13.3: s. 399-442.

POLÁK, Jiří, CARDA Antonín a MERUNKA Vojtěch. *Umění systémového návrhu: objektově orientovaná tvorba informačních systémů pomocí původní metody BORM*. 1. vyd. Praha: Grada, 2003, 195 s. ISBN 80-247-0424-2

ROMAN, D., a další. Web Service Modeling Ontology. *Applied Ontology 1.*, s. 77–106, 2005.

RUBIN, Kenneth S. a GOLDBERG, Adele. *Succeeding with objects: decision frameworks for project management*. Reading, Mass.: Addison-Wesley, c1995, xxiii, 542 p. ISBN 02-016-2878-3.

RUBIN, Kenneth S. a GOLDBERG, Adele. Object Behavior Analysis. *Communications of the ACM - Special issue on analysis and modeling in software development*. 1992, Sv. 35, 9.

SHAPIRO, Robert. *A Technical Comparison of XPDL, BPML and BPEL4WS*. [online]. 2002, s. 1-22 [Citace. 2012-07-01]. Dostupné z: http://www.w.bptrends.com/publicationfiles/Comparison%20of%20XPDL%20and %20BPML_BPEL%2012-8-02111.pdf.pdf

SHLAER, Sally a MELLOR, J., Stephen. *Object lifecycles: modeling the world in states*. Englewood Cliffs, N.J.: Yourdon Press, c1992, xiii, 251 p. ISBN 01-362-9940-7.

SCHMIDT, H.W. a REUSSNER, H.W. Generating Adapters for Concurrent Component Protocol Synchronisation. *In Proceedings of the Fifth IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS)*, Enschede, The Netherlands, March 2002. Kluwer Academic Publishers.

SMITH, Howard a FINGAR Peter. *Business process management: the third wave*. 1st ed. Tampa: Meghan-Kiffer Press, 2007, xvi, 292 s. ISBN 978-092-9652-344.

STEINBERG, Dave. *EMF: eclipse modeling framework*. 2nd ed. Upper Saddle River: Addison-Wesley, c2009, xxix, 704 s,. ISBN 978-0-321-33188-5

TAYLOR, David A. *Business engineering with object technology: modeling the world in states*. New York: Wiley, c1995, 187 p. ISBN 04-710-4521-7.

van der Aalst, Wil, M.,P., a další. From BPMN process models to BPEL web services. In: *Web Services, 2006. ICWS'06. International Conference on*. IEEE, 2006. p. 285-292.

van der Aalst, Wil, M.,P. Patterns and XPDL: a Critical Evaluation of the XML Process Definition Language. *BPM Center Report BPM-03-09*, BPMcenter.org, 2003

van der Aalst, Wil, M.,P., ter HOFSTEDE, Atrhur, H., M. a WESKE, Mathias. 2003. Business Process Management: a Survey. *Lecture Notes in Computer Science*. 2003, Sv. 2678.

VANÍČEK, Jiří. *Teoretické základy informatiky*. 1. vyd. Praha: Alfa Publishing, 2007, 431 s. ISBN 978-80-903962-4-1.

VERBEEK, H.,M.,W., vad der Aalst, Wil, M.,P. Analyzing BPEL processes using Petri nets. In D.C. Marinescu, editor, *Proceedings of 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management*, s. 59–78, June 2005.

W3C Glossary. *Web Services Glossary Working Group Note 11 February 2004*. [Online] 2004. [Citace: 2012-07-01]. Dostupné z: http://www.w3.org/TR/ws-gloss/

W3C WS-Architecture. *Web Services Architecture*. [Online] 2004. [Citace: 2011-11-12]. Dostupné z: http://www.w3.org/TR/ws-arch/

W3C WS-CDL. *Web Services Choreography Description Language Version 1.0*. [Online] 2004. [Citace: 2012-07-01]. Dostupné z: http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/

W3C WSCI. *Web Service Choreography Interface (WSCI) 1.0*. [Online]. 2002. [Citace: 2011-11-12]. Dostupné z: http://www.w3.org/TR/wsci

W3C WSCL. *Web Services Conversation Language (WSCL) 1.0*. [Online]. 2002. [Citace: 2012-07-01]. Dostupné z: http://www.w3.org/TR/wscl10/

W3C WSDL. *Web Services Description Language (WSDL) 1.1*. [Online]. 2001. [Citace: 2011-10-15]. Dostupné z: http://www.w3.org/TR/wsdl

W3C WSMO. *Web Service Modeling Ontology*. [Online] 2005. [Citace: 2012-02-23]. Dostupné z: http://www.w3.org/Submission/WSMO/

WEERWARANA, Sanjiva, a další. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. [s.l.] : Prentice Hall PTR, 2005. 456 s. ISBN 0-13-148874-0.

WFMC XPDL. *Workflow Management Coalition Workflow Standard Process Definition Interface - XML Process Definition Language*. [Online] 2008. [Citace: 2011-05-07]. Dostupné z: http://www.wfmc.org/View-document-details/WFMC-TC-1025-Oct-10-08-A-Final-XPDL-2.1-Specification.html

WSMO Choreography in WSMO. *Ontology-based Choreography and Orchestration of WSMO Services Final Draft 1 March 2005*. [Online] 2005. [Citace: 2012-07-01]. Dostupné z: http://www.wsmo.org/TR/d14/v0.1/

WSMO. *WSMO Web Service Discovery*. [Online] 2004. [Citace: 2011-11-12]. Dostupné z: http://www.wsmo.org/2004/d5/d5.1/v0.1/20041112/

WRIGHT, David, R. *Finite state machines: CSC215 Class Notes*. [Online]. 2005 [Citace 2012-09-21]. Dostupné z: www4.ncsu.edu/~drwrigh3/docs/courses/csc216/fsm-notes.pdf

YANG, Hongli, Z., X. a ZONGYAN, Q. A formal model of web service choreography description language (ws-cdl). *Technical report*, Department of Informatics, School of Mathematics, Peking University, China, January 2006.

Seznam obrázků a tabulek

Seznam obrázků

Seznam tabulek