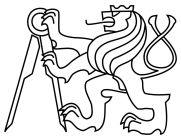
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Podpora standardu BPMN na platformě OpenPonk
Student:	Boris Anisimov
Vedoucí:	Ing. Robert Pergl, Ph.D.
Studijní program:	Informatika
Studijní obor:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

1. Seznamte se notací BPMN a jejím uplatnění v Enterprise Engineering (EE)
2. Seznamte se s platformou OpenPonk
3. Navrhněte architekturu modulu pro BPMN modelování v OpenPonk.
4. Implementujte prototyp BPMN modeláře zahrnující potřebnou podmnožinu BPMN pro EE
5. Navrhněte sadu verifikačních pravidel pro BPMN modely a implementujte tyto verifikace
6. Otestujte výsledek a proveďte vyhodnocení vaší práce z hlediska přínosů pro EE

Seznam odborné literatury

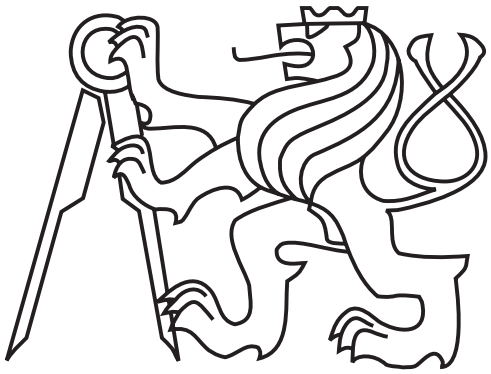
Silver, B. (2011). *BPMN Method and Style, 2nd Edition, with BPMN Implementer's Guide: A structured approach for business process modeling and implementation using BPMN 2.0*. Cody-Cassidy Press.

https://openponk.github.io

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. října 2017



Bakalářská práce

Podpora standardu BPMN na platformě OpenPonk

Boris Anisimov

Katedra softwarového inženýrství
Vedoucí práce: Ing. Robert Pergl, Ph.D.

15. května 2018

Poděkování

Děkuji svému vedoucímu práce, Ing. Robertu Perglovi, Ph.D., za podněty během tvorby bakalářské práce. Děkuji své rodině a přátelům za podporu nejen při tvorbě teto práce, ale i během celého studia.

Prohlášení

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

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

V Praze dne 15. května 2018 .....

Odkaz na tuto práci

Anisimov, Boris. *Podpora standardu BPMN na platformě OpenPonk*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Jedním z problémů při kreslení procesních diagramů je, že autor špatně spojí nebo chybně využije elementy. Tím způsobí nejasnosti a další práce s tímto diagramem ho může přivést ke špatnému výsledku. Tento problém se snaží vyřešit tato práce.

Literární rešerše se zabývá analýzou využití procesních diagramů při řízení podniku. Analýzou nastroje, který umožňuje kreslit takové diagramy. Dále analyzuje pravidla, pomoci kterých se nakreslené diagramy dají verifikovat.

Praktická část navázána na implementaci modulu pro platformu OpenPonk, který umožňuje kreslit procesní diagramy a ověřovat jejich správnost.

Klíčová slova modelovací plugin, diagram, BPMN, verifikace, OpenPonk, Pharo, enterprise engineering

Abstract

One of the problems in drawing process diagrams is that the author mistakenly combines or mistakes the elements. This causes confusion and further work with this diagram can lead to a poor result. The problem is trying to solve this problem.

Literary research deals with the analysis of the use of process diagrams in company management. An analysis of tools that allows you to draw such diagrams. It also analyzes the rules that can be used to verify the diagrams dialed.

The practical part followed the implementation of the OpenPonk module, which allows you to draw and verify the process diagrams.

Keywords modeling plugin, diagram, BPMN, verification, OpenPonk, Pharo, enterprise engineering

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Podniky vždy hledají cesty k úspěchu. Ty, které si zvolily cestu procesního řízení, musí vyřešit určtié problémy. Jedním z nich je správné zachycení podnikových procesů. Jedním z možných řešeni je modelování těchto procesů pomocí notace BPMN. Při modelování je potřeba znát formální pravidla použití notace. Ale i při dobré znalosti pravidel je možné udělat chybu při modelování. A přesně proto vznikla tato práce.

Výstupem této práce je plugin pro platformu OpenPonk. Plugin umožňuje kreslit procesních diagramů pomocí notace BPMN a dále je umí verifikovat. To usnadní podnikům práci při modelování procesů. Díky tomu, že plugin umí verifikovat diagramy, business analytici mají zaručeno, že diagram nebude obsahovat chyby.

Tato práce se zabývá analýzou notace BPMN a jejím použitím v rámci řízení business procesů. Dalším tématem je popsání verifikačních pravidel pro diagramy, vyvinuté pomocí notace BPMN.

Kapitola 1

Cíle a struktura práce

Cílem rešeršní části práce je seznámení se s notaci BPMN a jejím uplatněním v Enterprise Engineering, seznámení se s platformou OpenPonk a návrh verifikačních pravidel pro BPMN diagramy.

Cílem praktické části práce je implementace prototypu pluginu BPMN diagramu pro platformu OpenPonk, implementace verifikačních pravidel a testování hotového prototypu.

Práce pokračuje analýzou využiti procesního přístupu pro řízení podniku. Dalším bodem je analýza notace BPMN a její uplatnění pro procesní řízení. Dále jsou popána pravidla pro verifikaci BPMN diagramu.

Další částí je návrh pluginu pro OpenPonk. Na začátku je popsána architektura pluginu a dále uvedena samotná implementace. Na konci implementační částí je popsáno testování pluginu.

Poslední část je věnovaná diskuzi implementace a budoucímu vývoji.

Kapitola 2

Rešeršní část

2.1 Business process

Mistři procesního řízení Hammer a Champy vyzkoumali, že úspěch v podnikání přináší efektivní procesy, nikoliv samotné výrobky [1]. Většina výrobních firem používá procesní řízení, ale ostatní se často nezabývají svými business procesy. Firmy neustále hledají cestu optimalizace snížení nákladů z ekonomického hlediska. S použitím procesního řízení je možné snadno najít problémová místa a odstranit je bez zkrácení nákladů na personál [2]. Jedním z problémů v bankovní oblasti je, že zaměstnanci těchto firem nemají dostatečně definovanou svou pracovní roli. Ve výrobních firmách manažeři sledují zaměstnance a trestají je za nedodržovaní technologií. Naopak v nevýrobních firmách jsou činnosti zaměstnanců popsány jen obecně, což vede k tomu, že zaměstnanci mění náplň své práce. Zaměstnanci upravují svou činnost podle svých životních zkušeností, v důsledku se sníží efektivita a kvalita obsluhování zákazníků. Modifikace má hodně negativních důsledků, jako je například: rostoucí nespokojenost zákazníků, snížení možnosti kontroly ze strany manažera – to vede k růstu bankovního rizika [3, kapitola 1]. Firmy, které chtějí být efektivnější, musejí kontrolovat zaměstnance, aby neměnili naplň své pracovní činnosti.

Ve výzkumu firmy McKinsey Global Institute [4] je napsáno, že výkon práce v Rusku je stále na nízké úrovni ve všech oblastech včetně finanční. Zpracování požadavků v ruském finančnictví trvá déle než v jiných státech, i když víme, že je to jedna z nejvíce zautomatizovaných a nejbohatších oblastí. Dokonce v nejlepších bankách nejzákladnější operace trvají déle, až 2-6krát než v USA. Počet zaměstnanců na jednu operaci je 2-2,5krát vyšší, počet papírů je 1,5-3krát větší. Je chyba si myslet že zautomatizovanost je garancí vysokého výkonu.

Ve všech prozkoumaných oblastech je problém nedostatek přesného popsání činnosti zaměstnance, což vede k velké variabilitě procesů, snížení efektivity a kvality služeb. Z praxe víme, že nekoordinovaná práce vysoce kvalifikovaných zaměstnanců je méně efektivnější než koordinovaná práce obyčejné kvalifiko-

vaných zaměstnanců [3, kaptola 1.2].

Existují vynálezy, které hodně změnily náš život. Například – dopravní pás umožňuje zvýšit výkon práce v oblasti výroby. Dopravní pás umožňuje rozdělit celkovou práci na malé jednotky práce, které mohou být zpracovány jedním zaměstnancem.

Business proces reprezentuje virtuální dopravní pás z hlediska organizace práce, ne automatizace jednotlivých operací.

Pod pojmem business proces budeme rozumět souhrn prací, zaměřených na získávání reprodukovatelného výsledku [3, kapitola 1.5]. V této definici se mluví o reprodukovatelnosti, to také znamená, že každý výsledný kus může být jednoznačně identifikován. Počet výstupů procesů získaných v nějakém časovém intervalu může být spočten. Pokud firma vyrábí nějaké zboží, pak chce, aby každý kus byl totožný. A proto je potřeba aby všichni zaměstnanci pracovali podle standardů.

Business proces je technologie k dosažení plánovaného výsledku. Technologie je souhrn metod a nástrojů pro vyrábění produktů s popsanou kvalitou a malými náklady. Aby to bylo možné, technologie navrhuje rozložit práci na lehce vykonatelné části, zafixovat a popsat je. Požadavek reprodukovatelnosti výrobků je spojen s opakovatelnou činnosti zaměstnanců. Business proces je „technologie výroby“ služeb, které navzájem od materiálních výrobků ne vždy mají fyzickou podobu. Podobně – business proces je souhrn pravidel popisujících způsob a pořadí vykonání práce s cílem dosažení nejvyšší kvality a výkonu [3, kapitola 1.5].

2.2 Enterprise Engineering a BPM

Enterprise Engineering (EE) je souhrn znalostí, principů a postupů pro analýzu, návrh a řízení společnosti. V neustále se měnícím a nepředvídatelném konkurenčním prostředí odpovídá na otázku: „jak navrhovat a vylepšovat všechny prvky spojené s celkovým podnikem pomocí inženýrských a analytických metod a nástrojů k efektivnějšímu dosažení cílů [5]?“ Enterprise Engineering zkoumá každý aspekt podniku, včetně business procesů, informačních toků, materiálových toků a organizační struktury [6]. Enterprise Engineering je nadmnožinou nástrojů zahrnující hodně dalších podmnožin.

Jednou z podmnožin EE je Business Process Management (BPM). BPM je disciplína zahrnující jakoukoli kombinaci modelování, automatizace, provádění, řízení, měření a optimalizace toků podnikové činnosti [7]. BPM poskytuje firmám velkou hodnotu. Snižuje lidské chyby a chybnou komunikaci, digitalizuje manuální procesy a přenáší odpovědnost za dokončení procesu od lidí k programu [8].

2.3 Životní cyklus BPM

Aby BPM fungovalo korektně, musí se dodržovat jeho životní cyklus. Cyklus popisuje postup práce s business procesy od návrhu po měření výkonu.

Všechny činnosti životního cyklu BPM je možné rozdělit do skupin – design (návrh), modeling (modelování), execution (provedeni), monitoring (monitorování) a optimization (optimalizace). V různých zdrojích se některé kroky jmenuji jinak.

2.3.1 Návrh

Návrh je prvním krokem v životním cyklu BPM. Během tohoto kroku jsou procesy „as is“ zdokumentovány spolu s procesy „to be“. Pro znázornění je vytvořen diagram celého procesu, resp. procesů. Diagram musí obsahovat: tok úkolů, oznámení, eskalace, dohody o úrovni služeb a předávaní jednotlivých úkolů [9].

2.3.2 Modelování

Modelování business procesů je aktivita reprezentující procesy organizace v strukturální podobě. Nejpoužívanější formou je diagram nebo „workflow chart“, který mapuje tok činností v rámci procesu. Model business procesu hraje hlavní roli při vyhodnocovaní metrik a hledaní míst pro zlepšeni [10].

Ve své prací jsem se zaměřil na modelovací část životního cyklu.

2.3.3 Provedeni

Dalším krokem je provedení. Existují dvě hlavní možnosti: ruční implementace ve svých každodenních obchodních aktivitách nebo automatizace.

Manuální implementace zahrnuje předávání nových pokynů různým oddělením a zaměstnancům, kteří se budou podle nich řídit. To se hodí pokud je tým malý. Správce procesu může být přidělen ke sledování všech činností pro každý business proces.

Pokud tým obsahuje vice zaměstnanců, pak se může hodit automatizace pomocí Business Process Management Suite (BPMS) [11]. To nás přivede přímo do čtvrté etapy životního cyklu BPM: monitorování.

2.3.4 Monitorování

Pokud chcete neustále vylepšovat své podnikání (procesy) musíte si položit otázky – jsou plánované vylepšení opravdu vylepšení? Existují nějaké příležitosti k vylepšení?

Monitorování pomáhá měřit a analyzovat běh procesů, aby bylo možné identifikovat kritické problémy prostřednictvím získaných dat. Tímto způsobem je možné zlepšit rychlost, kvalitu a efektivitu procesů.

Vzhledem k tomu, že procesy velmi často procházejí mnoha odděleními, tak monitorování procesů je nejlépe prováděno jednou osobou nebo týmem, který má celkový pohled na celý proces. Daná osoba, resp. tým, se obvykle nazývá vlastníkem procesu [12].

2.3.5 Optimalizace

Pokud výsledky metrik nevypadají tak skvěle, co je potřeba dělat? Optimalizovat procesy.

Pro efektivní optimalizaci procesů společnosti je třeba shromáždit všechny informace o výkonu ve všech fázích životního cyklu BPM. Jedná se o shromáždění údajů o tom, co fungovalo a co nefungovalo. Od návrhu až po model. Hlavním cílem je zajistit, aby se zabránilo chybám, a aby se úspěchy opakovaly, minimalizovaly se náklady a maximalizovala účinnost.

Je potřeba se zaměřit spíše na ty klíčové oblasti, které potřebují modifikaci a ponechat vše, co běží, tak jak je. Výsledkem optimalizace je typicky úprava stávajících standardních provozních postupů pro efektivnější pracovní postup.

Stručně řečeno, životní cyklus BPM od návrhu až po optimalizaci zajišťuje bezproblémový pracovní proces ve společnosti. Aby se zabránilo chybám a byl zajištěn úspěšný chod společnosti, tak je potřeba životní cyklus neustále opakovat. Bez dobré znalosti pracovních postupů je praxe BPM nepoužitelná [10].

2.4 BPMS

Business Process Management System (BPMS) je software pro podporu konceptu BPM ve firmě. Systémy BPMS implementují koncept BPM pomoci softwaru [13].

BPMS považuje společnost za množinu procesů. BPMS nezkoumá práci jednotlivých oddělení, ale zkoumá sám proces prodeje, proces podpory zákazníků, proces řízení dodávek atd. Na základě tohoto probíhá práce na reengineeringu business procesů v BPMS.

BPMS je zaměřen především na zlepšení práce společnosti, na ziskovější činnost podniku prostřednictvím optimalizace a kontroly business procesů.

BPMS se dívá na zaměstnance jako na člověka zodpovědného za vykonaní své částí práce, ne jako na člověka zodpovědného za výsledek. Variabilita činnosti pracovníka je zde vyloučena. Zaměstnanec dělá pouze to, co mu systém umožní, ani více, ani méně [14].

Například zaměstnanec potřebuje vytvořit objednávku. Jaké jsou jeho činy?

Může dokument libovolně vyplnit, není-li proces přímo vydefinován.

- Můžete nejprve otevřít formulář objednávky, přidat zboží, specifikovat ceny, a potom určit zákazníka.

- Může nejprve vytvořit zákazníka, a pak jeho objednávku.

Stručně řečeno, v akcích uživatele existuje variabilita, tzn. zaměstnanec sám rozhodne, v jakem poradí vykoná jednotlivé akce.

Dále prozkoumáme, jak tento proces bude vypadat v rámci BPMS. Nejprve definujeme logiku práce a rozdělíme obchodní proces do postupných fází. V našem příkladu budou tři:

1. vytvořeni žádosti na zjištěni stavu konta;
2. kontrola žádosti;
3. výsledek žádosti:
    - pokud je schválena – výtisk žádosti,
    - pokud není schválena – informovaní zákazníka.

V systému každý uživatel pracuje pod svým loginem a heslem a vidí pouze svou roli v rámci obchodního procesu. V našem příkladu za vytváření žádosti odpovídá pracovník a za ověřování a schvalování nadřízený. Každý z nich vidí jen svou část formuláře, své úkoly a mohou vykonávat pouze své činnosti.

Důsledkem je, ze když zaměstnanec pošle žádost dále na kontrolu, tak se z něho odstraňuje zodpovědnost. Dál za žádost odpovídá nadřízený, který kontroluje žádost. Tímto způsobem můžeme přesněji kontrolovat zodpovědnosti.

Tento příklad ukazuje, že v BPMS vše závisí na kontextu. Všechny interakce s formami jsou zaměřeny na to, aby uživatel viděl pouze to, co je potřeba. A jen to, co potřebuje vidět v určité fázi, na základě kontextu procesu.

Ve skutečnosti v BPMS každý zaměstnanec pracuje jako na dopravním pásu. Každý úkol, každý business proces, kterého se zaměstnanec účastní, se stává odděleným pásem. A jako účastník tohoto procesu může zaměstnanec v rámci úkolu provádět pouze určité činnosti striktně omezené algoritmem plnění úkolu [14].

2.5 Historie BPMN

Specifikaci BPMN v 2001-2004 vyvíjela BPMI (Business Process Management Initiative). Notace popisovala jenom, jak mají vypadat grafické elementy, ale ne sémantiku řešení. Původně zkratka BPMN značila Business Process Management Notation. Později název se změnil na Business Process Model and Notation, tento název zdůrazňuje, že specifikace popisuje jak grafickou část, tak i execution model. V roce 2004 se vývojem specifikace zabývala OMG (Object Management Group), pak v únoru 2006 OMG zveřejnila tuto specifikaci jako vlastní. V roce 2011 byla zveřejněna aktuální verze specifikace 2.0 [15].

2.6 Specifikace 2.0

Notace BPMN 2.0 je nástrojem pro grafické modelování business procesů a zároveň popisuje, jak se má generovat „execution model“ z diagramu. Hlavní výhoda je v tom, že BPMN je orientována na širokou škálu odborníků zapojených do popisování a automatizovaní business procesů. Notace umožňuje pracovat s modely na různých úrovních abstrakce – od koncepčních modelů interakce účastníků business procesů po technická schémata, které popisují běh procesů [16].

Notace má status průmyslového standardu [17]. To znamená že používání BPMN není povinné, ale díky velké podpoře notace a reputaci OMG, se používání BPMN šíří. V současné době většina modelovacích systémů a systémů řízení business procesů má podporu standardu 2.0 [18].

BPMN má svůj syntax (popisování označení elementů a pravidla kombinování elementů) a sémantiku (popisování interpretace modelů a jejich elementů). Sémantika BPMN je velmi důležitá, protože popisuje, jak se chovají grafické elementy ze schématu v realitě.

Specifikace 2.0 obsahuje tři části. První popisuje grafické elementy a pravidla, jejich použití při modelování business procesu. V druhé části specifikace je formalizován popis grafických elementů a vztahy mezi nimi, to umožňuje přenášet diagramy mezi různými modelovacími nástroji a neztrácet žádnou informaci. V třetí části je popsána sémantika vykonání grafických elementů, to umožňuje export diagramu do kódu pro vykonání business procesů (generovaní „execution modelu“) [19].

2.7 Cíle BPMN

Notace BPMN byla vymyšlena jako jednoduchý způsob vizualizace business procesů. Do vzniku BPMN neexistoval všeobecný standard vizualizace a každá společnost používala jiné nástroje. BPMN přišel s myšlenkou, že by nemělo záviset, jaký byl použit nástroj pro návrh modelů, výsledek vždy bude stejný a jasný komukoli.

Hlavním cílem vývojářů BPMN bylo vytvořit standard, který bude jasný všem business uživatelům. Business analytikovi, který vytváří a zlepšuje procesy, vývojářům, kteří naplňují procesy, manažerům, kteří kontrolují běh procesů. BPMN se stala spojovacím článkem mezi business uživateli, kteří v jasné formě mohou popsat své požadavky a IT specialisty, kteří vyvíjí tyto požadavky pro IS [17].

Dalším faktorem, který vedl vývoj BPMN, je to že historické modely business procesů, které vyvinuli business analytici byly technicky odděleny od procesní reprezentace vyžadované systémem pro implementaci a provedení těchto procesů. Bylo potřeba ručně překládat originální procesní modely do „execution modelů“. Takové překlady obsahovaly chyby a byly složité pro majitele

procesu. Také bylo těžko změřit metriky takového procesu. Klíčovým cílem pro vývojáře BPMN bylo vytvořit most mezi grafickým modelem a „execution modelem“ [20].

2.8 Čím je dobrá notace BPMN

Notace BPMN dovoluje začít vývoj od abstraktního analytického modelu a doplňovat jej malými detaily, tím zvyšuje přesnost business procesu. Na konci bude úplně popsáný business proces [21].

Ještě jedna výhoda je v tom, že model je možné navrhnout s minimálním programováním. Velkou část práce udělá business analytik bez IT specialisty [22].

BPMN zmenšuje rozdíl mezi modely „as is“ (jak je to teď) a „to be“ (jak musí to být). „Execution model“ dává nejen možnost validovat model business procesů, ale také vyzkoušet, jak se bude proces chovat v reálném světě. Takové testy umožňují najít úzká místa a odstranit je.

V realitě to dovoluje jít od modelu „as it“ k modelu „to be“ malými evolučními změnami a měřit metriky procesu. Z toho důvodu získané modely „to be“ mají být objektivní, protože se opírají o výsledky testů [3].

2.9 Oblasti použití BPMN

BPMN je určena pro popsání:

- pořadí vykonání prací tvořící business proces,
- data flow mezi operací,
- message flow mezi procesy,
- spolupráce datových objektů v rámci procesu [3, kapitola 2.2].

2.9.1 Uplatnění BPMN v BPM

Jedním z bodů životního cyklu BPM je modelování procesu. Existuje velké množství technik na modelování procesu [23], ale nejpoužívanější je BPMN. Díky své jednoduchosti a užitečnosti BPMN je teď de facto průmyslový standard [24].

Hodně BPMS nástrojů maji BPMN modeler, který podporuje verzi 2.0, ale většina z nich nemá podporu generovaní „execution modelu“. V tuto chvíli, v roce 2018, jenom 3 nástroje mají tuto funkcionalitu – Camunda BPM[1], jBPM[2] a Activiti[3] [25].

[1] `camunda.com`
[2] `www.jbpm.org`
[3] `www.activiti.org`

Jak už jsem napsal výše – BPMN nedefinuje nejen grafické elementy, ale i „execution model“ na základě kterého pracuje BPMS. Vygenerováním z diagramu vzniká „execution model“, a ten se dále spouští, a pak je možné změřit metriky procesu. Na základě výsledků lze najít úzká místa a optimalizovat proces.

Z toho plyne že BPMN je široce používaná notace, která umožňuje nejen nakreslit grafický model procesu, ale i z něj vygenerovat „execution model“.

Ve své prací jsem implementoval jednu funkčnost BPMS – návrh grafického modelu business procesu.

2.10 Základní elementy BPMN

V této práci se omezím na tři základní kategorie grafických elementů:

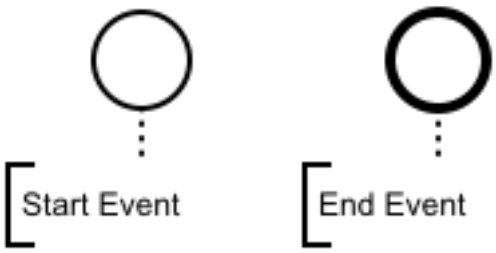
- tokové objekty,
- spojovací objekty,
- kontexty.

2.10.1 Tokové objekty

Tokové objekty jsou událost (event), činnost (activity) a brána (gateway).

Události se používají pro vícero účelů. První – ukázání časového úseku vykonání práce. Druhý – omezení délky trvání operace. Třetí – popisování reakcí na změnu stavu externích k operaci objektů.

Existují dva základní typy událostí – start event a end event. Start event představuje startovací bod procesu. End event představuje výsledek události nebo procesu. Na obrázku 2.1 je vidět start event a end event.

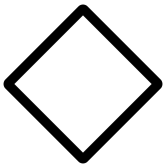


Obrázek 2.1: Start a end event

Činnost označuje jednotku práce, která mění stav procesu. Na obrázku 2.2 je vidět aktivita.

Brána umožňuje větvení nebo slučování toků procesu na základě podmínky. Na obrázku 2.3 je vidět gateway.

Obrázek 2.2: Activity



Obrázek 2.3: Gateway

Obrázek 2.4: Pool a dvě lane

2.10.2 Kontexty

Pro označování kontextu se používá bazén (pool) a dráha (lane).

Bazén označuje hranice procesu. Pool není třeba vždy kreslit. Pokud se na diagramu nevyskytuje, tak ani není potřeba, protože diagram je sám o sobě jednoduchý. Není tedy třeba oddělovat jeho různé části do několika celků.

Pool je rozdělen na dráhy (lanes). Dráhy slouží ke seskupování operací a mají název.

Na obrázku 2.4 můžete vidět pool a dvě lanes.

2.10.3 Spojovací objekty

Spojovacími objekty jsou sekvenční tok (sequence flow) a tok zpráv (message flow). Tyto objekty slouží pro propojení ostatních elementů.

Sekvenční tok spojuje události, činnosti a brány do řetězce. Určuje také pořadí vykonání.

Tok zpráv popisuje strukturu vyměňování zpráv mezi procesy. Nelze ho využít v rámci jednoho kontextu.

2.11 Verifikační pravidla

Pro navržení správného a jasného diagramu musíme dodržovat pravidla. OMG neposkytuje seznam pravidel pro BPMN diagramy [26]. Musel jsem taková pravidla najít. Ve knize „Method and Style“ je uveden seznam verifikačních pravidel pro „non-executable“ a „executable modely“. V této práci zúžím seznam pravidel a omezím se jen na některá. Budu pracovat s pravidlami pro sequence flow, start event, end event a gateway.

2.11.1 Sequence flow

- Sequence flow musí spojovat elementy (activity, gateway nebo event). Nikdy nesmí být nespojena.
- Všechny elementy, kromě start event, musí mít příchozí flow.
- Všechny elementy, kromě end event, musí mít odchozí flow.
- Activity nebo gateway má mít maximálně jeden default flow.

2.11.2 Start event

- Start event nemá mít příchozí flow.

2.11.3 End event

- End event nemá mít odchozí flow.

2.11.4 Gateway

- Rozdělovací gateway musí mít více než jeden východ.

Dále bych chtěl popsat pravidla, která jsou logická, ale nejsou uvedena v seznamu výše.

Sequence flow nemůže spojovat jenom start a end eventy. To pravidlo nedovoluje mít na diagramu jenom start a end eventy, protože takový diagram by nedával smysl.

Na diagramu vždy musí být start event, end event a minimálně jeden activity element. Díky tomu diagram bude mít alespoň jednu akci a bude dávat smysl.

2.12 Příklad použití

Na obrázku 2.5 můžeme vidět jednoduchý příklad použití BPMN, který popisuje Order proces. Na začátku dostaneme Order. Pak se rozhodneme, jestli ho budeme dál zpracovávat. Za rozhodování odpovídá gateway s X. Čára s malou čárkou se nazývá *default flow*, je to ta flow, kterou půjde proces, pokud ani jedna z podmínek nebude splněna. Dál z „Fill Order“ vede čára do gateway s plusem. Ten gateway rozdělí běh procesu na dvě paralelní flow. Další gateway čeká na obě flow. Další gateway s X spojí dvě flow, ale víme z prvního gateway že flow půjde jen jedním směrem. Pak Order bude uzavřen a proces skončí.

Jako základ jsem použil příklad z webu [27].

2.13 Existující řešení

V současné době existuje hodně nástrojů umožňující práci s BPMN. Ale ne všechny jsou cross-platformové a nepodporují verifikaci diagramu. Většina z nich jsou placené. Existuje několik zajímavých řešení. Například existují BPMN editory pro chytré telefony (iOS, Android a Windows Phone), ale jsou placené. Také existuje několik cloud řešení a browser editorů.

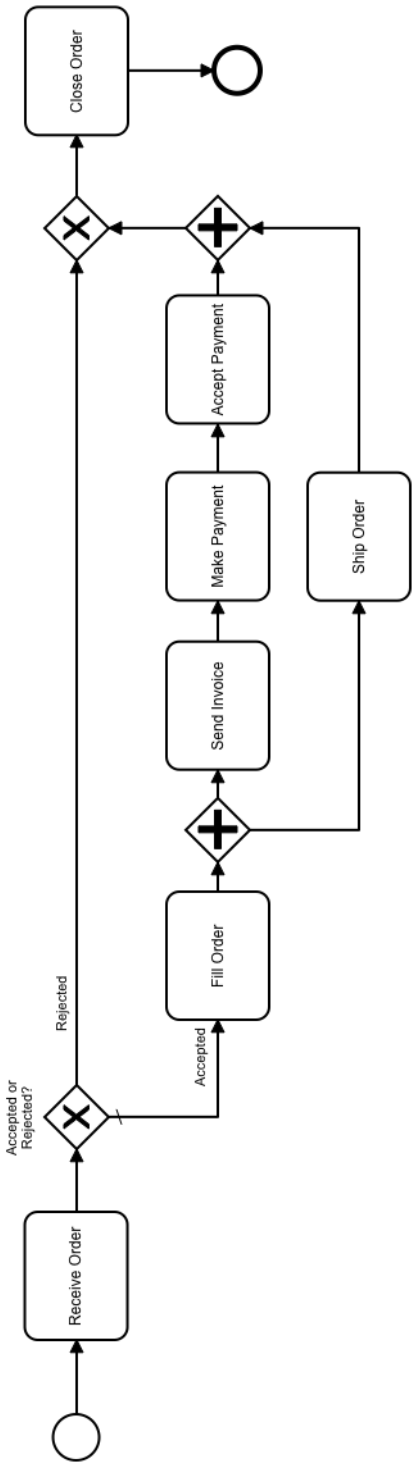
Při rešerši jsem našel zajímavý editor bpmn.io[4] – webový editor. Tento editor je úplně zdarma, má minimalistický design a podporuje nejen BPMN. Bohužel neumí ale verifikovat diagram, ani generovat „execution model“. Také velká výhoda bpmn.io je, že je to open source. Kdokoli si může stáhnout kód, upravit ho a spustit na svém počítači.

V tabulce 2.1 je uvedeno několik nástrojů, které podporují práci s BPMN. Tato informace je převzatá z [18].

2.14 OpenPonk

OpenPonk je experimentální platforma pro konceptuální modelování, vyvíjená na Pharo. Pharo je kombinací čistě objektového jazyka programování založeného na Smalltalku, virtuálního stroje a vývojového prostředí. Jedna z výhod OpenPonk je, že můžeme jednoduše rozšířit funkcionalitu platformy. OpenPonk vznikl v roce 2014 na katedře softwarového inženýrství Fakulty informačních technologií jako týmový projekt [28]. Platforma je stále ve stavu „early development“, a proto obsahuje ještě sadu bugů.

[4] `https://bpmn.io`



Obrázek 2.5: Order flow

Nazev	Autor	OS	Licence
bpmn.io	Camunda Services GmbH	Cloud, Web	Open Source, Free
yEd	yWorks	Windows, Mac, Linux/Unix	Free
Enterprise Architect	Sparx Systems	Windows, Linux, Mac	Proprietary
Bizagi BPM Suite	Bizagi	Windows	Proprietary
LucidChart	Lucid Software Inc	Web	Free, Proprietary
AuraPortal	AuraPortal	Windows	Free, Proprietary
BeePMN	ESTECO SpA	Cloud	Free, Proprietary
Bonita BPM	Bonitasoft	Windows, Linux, Mac	Open Source, Free

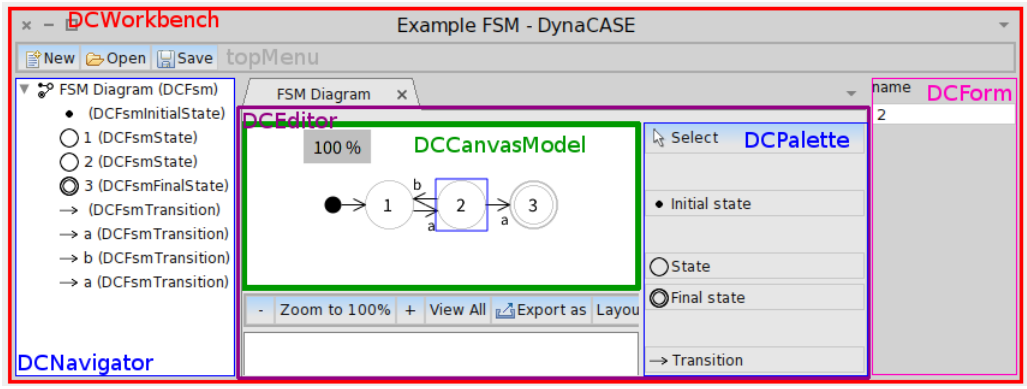
Tabulka 2.1: Seznam modelovacích nástrojů

Na obrázku 2.6 je ukázáno hlavní okno s editorem OpenPonk. Obrázek byl převzat z webu OpenPonk, je z roku 2016. Na obrázku je použita předpona *DC* po třídy, ale v současné době OpenPonk má předponu *OP*. Grafický editor se skládá z několika částí. *DCNavigator* obsahuje stromovou reprezentaci modelu. *DCForm* umožňuje přidat políčka pro atributy nějakého elementu. *DCPalette* ukazuje, jaké elementy se mohou použit při modelování. *DCCanvasModel* je plátno pro diagram. *DCEditor* je samotny editor. *DCWorkbench* spojuje všechny elementy. Kazdou z těchto tříd je možné rozšířit v rámci pluginu [29].

OpenPonk je vyvíjen pomocí vzoru MVC (model-view-controller). Model reprezentuje data, view representuje uživatelské rozhraní. Model a view jsou nezávislé části. Controller spojuje model a view a určí co z modelu se má zobrazit uživateli. Model reprezentuje jak element, ale i hranu – spojku mezi elementy. Pro každý grafický element existuje controller a celý diagram také má svůj controller.

Existuje možnost poslouchání události které se staly s modelem. Jakýkoliv objekt se může přihlásit k odběru události, jako je například změna nebo smazání. Když se objekt modelu změní, pak se pošle oznámení všem odběratelům. Tímto způsobem může být změna reprezentována uživateli [28].

Existuji dvě základní třídy controlleru elementů, od které se má dědit – *OPElementController* a *OPRelationshipController*. První controller se využívá



Obrázek 2.6: Grafické rozhraní OpenPonk

pro uzlové elementy, druhý slouží jako spojka mezi elementy. Také existuje jediný controller pro cely diagram – *OPDiagramController*, který popisuje chování diagramu a je zodpovědný za jeho inicializaci. Dále existuje controller pro celý plugin. Ten vrací všechno, co je potřeba pro start pluginu – controller diagramy, model diagramy a název pluginu [30].

Každý controller elementu má taky v sobě view, které se vrací v *figure*, která reprezentuje view z MVC [30].

2.14.1 Přidávaní elementu na plátno

OpenPonk má dobrý systém přidávaní elementu na plátno. Daný systém je možné přímo použít pro aplikaci verifikačních pravidel.

Existuje třída *OPCreationTool*, která uchovává factory pro vytváření elementů a řídí jejich samotné vytváření. Na začátek je nutno zavolat *newCreationTool: factory:* ve třídě *OPPalette*. V prvním parametru je název elementu, v druhém je blok, ve kterém je nutno vrátit controller pro nový element. Při vkládaní nového elementu se vždy ten element umístí do cíle (samo plátno, nebo jiný element). Chcete-li úspěšně přidat nový prvek, musí ho cíl přijmout, to je řízeno metodou cílového controlleru *canBeTargetFor: aController*. Ve výchozím nastavení nikdo nepřijme nic. Pokud cíl přijme element, musí také implementovat metodu *addAsTargetFor: aController*, protože cíl poskytuje kontext v rámci nového prvku. Na obrázku 2.7 je podrobnější popsaní volaní při vytváření elementu. Obrázek byl převzat z [31].

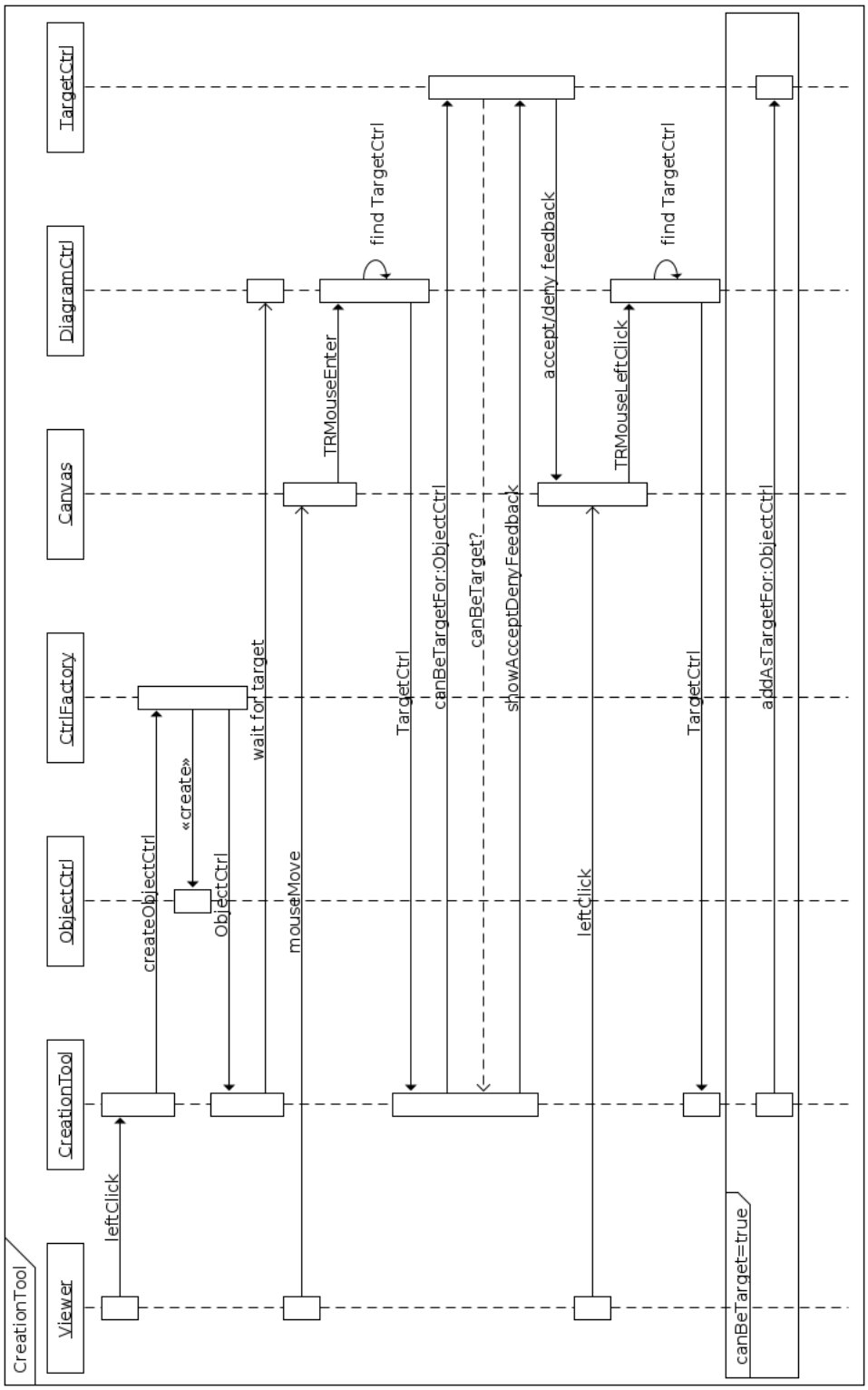
2.14.2 Architektura modulu

Dědění Při vývoji jsem stavěl nad platformou OpenPonk, která využívá architektonický stylu MVC. Během své práce jsem tento styl i nadále dodržoval. Dále popíši základní třídy modelů. Při implementaci využívám třídy modelů, které musí dědit od *OPMElement*. Třídy modelů budou zodpovědné za in-

terní uchováváni diagramu. Každý grafický element má svou třídu modelu. Například model elementu start event je realizován pomoci třídy *BPMNStartEvent*, která dědi od *BPMNEvent* kvůli přehlednosti. Je také potřeba mít cotrollery pro každý grafický element. Controller pracuje s modely a jejich grafickou reprezentací. Controller je dále zodpovědný za propojení elementů mezi sebou a zrušení vazby.

Controller diagramy Jednou ze základních tříd je controller diagramu, který zodpovídá za celý diagram. Ten je zodpovědný za prvotní inicializaci modulu, musí tedy umět přidávat elementy na plátno a popisuje výchozí rozvržení diagramu. V metodě *initializePalette:* controller vrátí factory, která bude vyrábět každý element diagramu. Tato factory bude využita systémem pro přidávaní elementu na plátno.

Navigátor Dalším bodem je navigátor, který obsahuje stromovou reprezentaci diagramu. Třída *OPNavigator* je náš navigátor. Tento element je možné nastavit v pluginu, ale pokud navigátor nebude nastaven, tak OpenPonk použije výchozí implementaci [32]. Ve své práci jsem nevyužil navigátor.



Obrázek 2.7: Sekvenční diagram vytváření elementu a přidávaní na plátno
20

Kapitola 3

Implementace

3.1 Struktura tříd

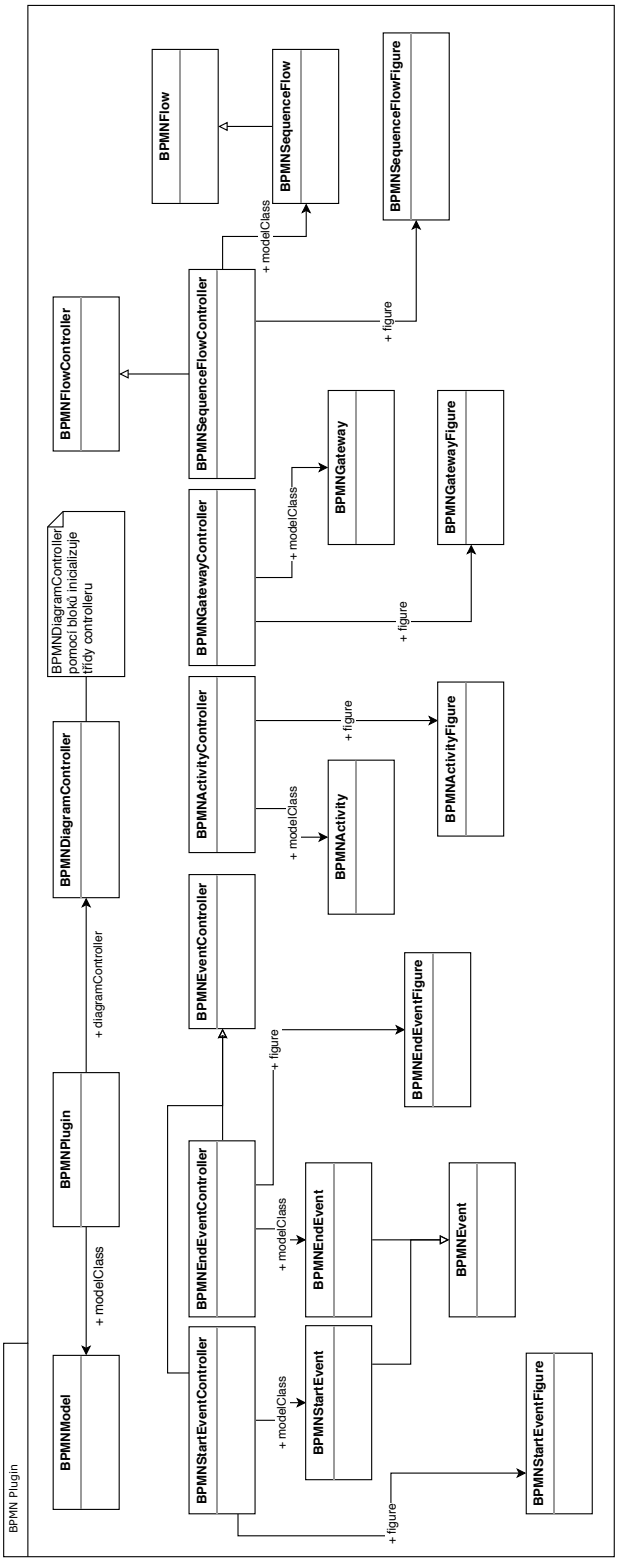
Jak už bylo popsáno výše – OpenPonk nabízí základní třídy pro práci s platformou. Na začátek chci uvést graf balíčku BPMN pluginu. Jak je vidět na obrázku 3.1, při návrhu jsem použil dědění. Má to výhodu – dělení logiky na menší části. Společná logika pro event a flow je implementována v nadtřídách. Podtřídy controleru pro eventy a flow obsahují už jenom logiku pro jednotlivé typy elementů. Například start a stop eventy mají různá pravidla pro spojování pomocí sequence flow (pravidla jsou v 2.11).

Každý controller implementuje metody, které jsou využity systémem pro přidávaní elementu na plátno. Díky dědění jsem přemístil společnou logiku, která rozhoduje, jestli budou elementy spojeny, do nadtříd. Nadtřída se zeptá svého potomka na možnost spojení a na základě toho vrátí rozhodnutí systému. Metody pro rozhodnutí jsou *canBeSourceFor: aController* a *canBeTargetFor: aController*, vracejí *true* nebo *false*. Například *BPMNEndEventControlelr* má v metodě *canBeTargetFor: aController* rozhodování *aController modelClass = BPMNSequenceFlow*. Na základě typu modelu controller vrátí hodnotu. Výhodou takové implementace je flexibilita.

3.2 Startovací bod pluginu

Pro spuštění pluginu je potřeba mít potomka třídy *OPPlugin*. Tato třída vrací název modulu, který bude v seznamu při startu OpenPonk, hlavní třídu controlleru diagramu a hlavní model diagramu. Při startu OpenPonk vyhledá a projde všechny nalezené potomky *OPPlugin*. Třída mého pluginu se nazývá *BPMNOPPlugin*.

Při procházení všech pluginů se registrují modely diagramu. Můj plugin vrací třídu *BPMNModel*. Tato třída reprezentuje model celého diagramu. Registraci modelu provádí *OPProjectBrowser*. Tato třída reprezentuje okno, které se ukazuje při startu OpenPonk. Dal při kliknutí na položku v menu



Obrázek 3.1: Diagram tříd BPMN pluginu

OPProjectBrowser vyhledá pomocí modelu plugin a přidá ho pro další zpracování.

Dalším krokem je načítání pluginu třídou *OPWorkbench*. Tato třída je zodpovědná za celé okno s editorem a obsahuje další části jako *OPForm*, *OPPalette* a *OPEditor*. Ve své práci jsem rozšířil chovaní *OPForm*. Také jsem musel naimplementovat factory pro *OPPalette*, která je částí systému pro přidávaní elementu na plátno.

3.3 Grafické uživatelské rozhraní

3.3.1 Ikony

Dalším problémem, se kterým jsem se setkal při vývoji pluginu, byly ikony pro elementy. Existuje několik míst, kde se mají ukazovat jednotlivé ikony. Jedním z nich je například pravá strana editoru nebo samo plátno editoru.

Plugin musí poskytovat platformě ikony pro pravou stranu editoru (*OPPalette*) ve formátu .png a encodované pomocí base64. Vytvořil jsem třídu *BPMNIcon* která poskytuje metody, které vrací ikony v požadovaném formátu. Ikony se načítají ve třídě *BPMNDiagramController* při inicializaci factory pro přidávaní elementu na plátno. Pak je seznam grafických elementů zobrazen na pravé straně editoru. Při kliknutí na ikonu se spustí blok, který byl zapsán do *factory:* palety.

Na obrázku 3.2 je vidět celou paletu s ikony.

Obrázek 3.2: Paleta s ikony

Ikony použité v pluginu jsem nakreslil v editoru obrázků a encodovaní jsem prováděl na webu `base64-image.de`. Vzhled nakreslených grafických elementů je v souladu se specifikací BPMN 2.0.

3.3.2 View

Dalším bodem je view. Controller jednotlivého elementu obsahuje metodu *createFigure*, která vrací třídu pro zobrazení elementu. Každý element má svou figure třídu, například start event má *BPMNStartEventFigure*. Figure třída definuje layout, popisuje chování view a může obsahovat další view (například štítek). Třída má metodu *baseShape*, která vrací základní tvar pro view, například *RTRoundedBox* pro obdélník nebo *RTEllipse* pro elipsu. Nebylo těžké nakreslit start a end event a activity. Tyto elementy jsou jenom základní formy.

Při implementaci jsem narazil na problém vykreslování gateway elementů. Existuje omezení platformy, které neumožňuje měnit velikost view, pokud je vykresleno pomocí Bézierovy křivky. Proto jsem musel použít *RTBox*, který jsem otočil o 45 stupňů. To každopádně daný problém ale nevyřešilo. Při zvětšování gateway se view chovalo jinak, než jsem čekal. V době psaní této práce jsem nenašel jiné řešení a zakázal jsem zvětšovaní gateway. V dalších verzích pluginu musí být tento bug odstraněn.

3.3.3 Štítek pro activity

Aby bylo možné napsat název tasku pro activity, tak je mu potřeba přidat šítek (label). Label se nachází uvnitř activity elementu. Každá třída pro figure dědí od třídy *RTAbstractMultiElement*. Je to vnitřní třída balíku, který kreslí UI. Tato třída reprezentuje view, které může být složeno z několika grafických elementů. Nadtřída má několik metod pro nastavení view. Použil jsem *createOwnedElements* na vytváření štítku a nastavovaní jeho pozice. Když uživatel napíše něco do políčka pro název elementu (*OPForm*), tak se daný test propíše do štítku. V případě změny obsahu štítku je nutné vyvolat překreslení celého view zavoláním metody *update*. Když se změní název modelu, pošle se událost o změně. Controller elementu tuto událost příjme a zavolá metodu *refreshFigure*, ve které bude zavolaná metoda *update* na figure.

3.4 Controllery

Nejdůležitější funkcionalitou controlleru elementu je rozhodovaní, jestli jiné elementy mohou být spojeny nebo vložené do sebe. Funkcionalita je částí systému pro přidávaní elementu na plátno.

Controller má ještě jednu užitečnou funkcionalitu – metodu *buildEditorForm:*. Tato metoda může vytvářet různá políčka pro *OPForm*. Políčka se budou nacházet v pravé části editoru. Existuje několik variant, co může se zobrazit – checkbox, droplist, label, text a textInput. Tyto elementy mohou být využity pro nastavování příznaků elementů. Ve své práci jsem použil checkbox pro nastavování default flow u gateway. Každý element formy může mít

nastaveného observera, který sleduje na jeho změny. V případě vyslání notifikace se vyková blok kódu definovaný programátorem.

3.5 Verifikace

Tlačítko *Validate* spouští verifikaci celého diagramu. Toto tlačítko se nachází v horním baru. Aby se tam objevilo, tak jsem naimplementoval statickou metodu *toolbarFor:* ve třídě *BPMNPlugin* a použil jsem pragma *<dcEditorToolbarMenu: #BPMNOPPlugin >*. Díky tomu OpenPonk pozná, že musí použít můj kód pro vytváření menu [33]. Poté při stisknutí tlačítka *Validate* se spouští blok *action:*, který byl předán při vytváření tlačítka. Blok může obsahovat libovolný kód. V našem případě blok spouští verifikační testy.

Třída *BPMNValidator* provádí verifikaci diagramu. Při inicializaci přijímá model diagramu a spouští jednotlivé testy. Každá metoda třídy reprezentuje validaci jednotlivých elementů. Například metoda *validate* spouští všechny testy a *validateSequenceFlow* spouští testy jenom pro sequence flow. Verifikační metody vrací *true* nebo *false* na základě nalezení porušení verifikačního pravidla. To se také hodí na testovaní správnosti verifikací.

Některé situace jsou pokryté pomocí metod controlleru *canBeSourceFor:* a *canBeTargetFor:*, protože uživatel hned vidí, jestli může spojit elementy nebo přidat element. Pokud je tato funkcionalita povolena, tak element bude svítit zeleně. V opačném případě červeně. Pokud akce není povolena, nepůjde přidat element nebo je spojit.

Kapitola 4

Testování

V této části své práce popíši, jak jsem testoval výsledný plugin. Také zhodnotím svou práci z hlediska přínosů pro EE.

4.1 Testování pluginu

OpenPonk poskytuje základní třídu pro testování *OPTestCase*. Tato třída dědí od základní testovací třídy Pharo *TestCase* a nic nedoplňuje. Při návrhu testů jsem použil výchozí metodu pro nastavení dat před testem *setUp*. Tato metoda se vždy volá před začátkem testování.

Všechny testovací soubory se nachází ve složce Tests. Naimplementoval jsem několik tříd, které testují komponenty pluginu. Začal jsem s testováním jednoduchých verifikačních pravidel. První testovací case jsem navrhnul pro pravidlo start event – může mít jenom odchozí sequence flow. Za zmíněný test odpovídá třída *BPMNTestStartEventFlow*. Na začátku testu je potřeba vytvořit mini diagram. Vytváření diagramů se provádí v metodě *setUp*. Každá testovací třída připravuje pro své testy diagram. Na testování potřebujeme start event a k tomu budeme přidávat sequence flow. Test se provádí pomocí příkazu *assert*. Díky tomu, že jsem naimplementoval metody *canBeSourceFor:* a *canBeTargetFor:*, je můžeme v testech zavolat a získat potřebné informace.

Bohužel testování některých verifikačních pravidel vyžaduje GUI – editor. Bez nastavení editoru třída *BPMNDiagramController* nemůže přidávat controllery do diagramu. Proto jsem musel navrhnout testování nejen verifikačních pravidel. Pro ruční testování si stačí otevřít OpenPonk s pluginem BPMN, přidat elementy na plochu a zkoušet přidávat a spojovat různé prvky.

Dále jsem naimplementoval testování základních funkčností tříd. Testoval jsem spojování modelů. Třída *BPMNTestModelConnection* testuje spojování elementů start event s end eventem pomocí sequence flow. Před začátkem testu je potřeba vytvořit modely pro každý z elementů a spojit je. Dalším krokem je ověření, jestli se modely doopravdy spojily. Model, který reprezentuje sequence

flow, má mít jako *source* start event a jako *target* end event. V rámci testů je pomocí příkazu *assert* tato podmínka ověřena.

Dále krátce popisuji testovací třídy.

- **BPMNTestStartEventFlow** implementuje testování pravidla – start event může mít jenom odchozí sequence flow.
- **BPMNTestEndEventFlow** implementuje testování pravidla – end event může mít jenom příchozí sequence flow.
- **BPMNTestModelConnection** implementuje testování správného spojení modelů.
- **BPMNTestValidationHasStartEnd** implementuje testování metody *validateDiagramHasStartEnd* třídy *BPMNValidator*.
- **BPMNTestValidationOnlyStartEnd** implementuje testování metody *validateOnlyStartEndEvents* třídy *BPMNValidator*.
- **BPMNTestValidationUnconnectedFlow** implementuje testování metody *validateSequenceFlow* třídy *BPMNValidator*.

Pro spuštění testů stačí v Systém Browseru vybrat balíček BPMN, najít složku „Tests“ a zmáčknout kulaté tlačítko před názvem testovací třídy. Po úspěšném proběhnutí testů bude tlačítko svítit zeleně. V opačném případě se zbarví do červena.

4.2 Přínosy pro Enterprise Engineering

Pro úspěšné procesní řízeni nejen společnosti je důležité mít k dispozici přesné a správné informace. Na začátku životního cyklu BPM je potřeba správně pochopit chování podniku a na základě toho navrhnout diagram procesu. Pro návrh diagramu je potřeba znát pravidla nastroje, který byl použit. V mém případě se jedná o BPMN. Všichni jsme lidi, a jak se říká: „chybovat je lidské“. Proto také dnes existuje možnost automatizovaně hledat chyby. Prototyp pluginu, který vznikl v rámci teto práce, umí verifikovat navržený BPMN diagram podle pravidel, které jsem popsal výše v sekci 2.11. To dává business analytikům velký přínos. Ale stále business uživatelé musí vědět, jak pracovat s BPMN. V knize *BPMN Method and Style* autor popsal, jak navrhnout správný diagram, a jak pracovat s „executable modely“. Tato knížka je základem pro ty, kteří pracuji s BPMN.

Doufám, že nám počítače budou více pomáhat při návrhu (nejen) složitých procesních diagramů.

Kapitola 5

Shrnutí implementaci a diskuze

Na konci práce bych chtěl popsat proces vývoje BPMN pluginu krok za krokem.

Můj vedoucí práce mi poradil sepsat, co jsem dělal každý týden vývoje. Průběžně jsem si dělal poznámky, které níže prezentuji.

V předmětu BI-OMO jsem se seznámil s prostředím Pharo. Díky tomu jsem rychle pochopil kód OpenPonk. Jedním z větších problémů bylo pochopit jednotlivá volání kódu – co, a kdy se volá. Na začátek jsem prošel kód existujícího pluginu FSM. Je to plugin, který simuluje konečné automaty. Dále jsem začal pomalu dopisovat části svého pluginu. Začal jsem úplně od základních částí – názvu a třídy *BPMNDiagramController*. Pak jsem implementoval jednotlivé elementy. Když jsem už měl základní strukturu tříd, začal jsem pracovat nad vzhledem elementů. Vygeneroval jsem ikony pro pravou část editoru.

Dále jsem naimplementoval *Figure* pro každý element. Nečekal jsem, že se setkám s tak velkým problémem při kreslení gateway. Bohužel jsem ho dodnes nevyřešil.

Dále jsem začal s implementací logiky komunikace elementů. Naimplementoval jsem metody pro přidávání a spojování elementů. Také jsem se rozhodl, že částečně naimplementuji verifikační pravidla. Dále musel jsem upravit třídy modelů, abych je mohl používat bez větších potíží. Začal jsem hledat, jakým způsobem se budou spouštět verifikační testy. Nejlepším řešením bylo navrhnout třídu, která se o vše postará. A tak vznikla třída *BPMNValidator*.

Dále bylo potřeba spustit testy. V dokumentaci jsem našel, jak přidat tlačítko do horního baru. Učinil jsem tak a dnes to tlačítko spouští testy. Dalším krokem byl přepis verifikačních pravidel do kódu.

Dle zadání jsem měl otestovat výsledek. Začal jsem psát jednotlivé testy pro každou komponentu pluginu.

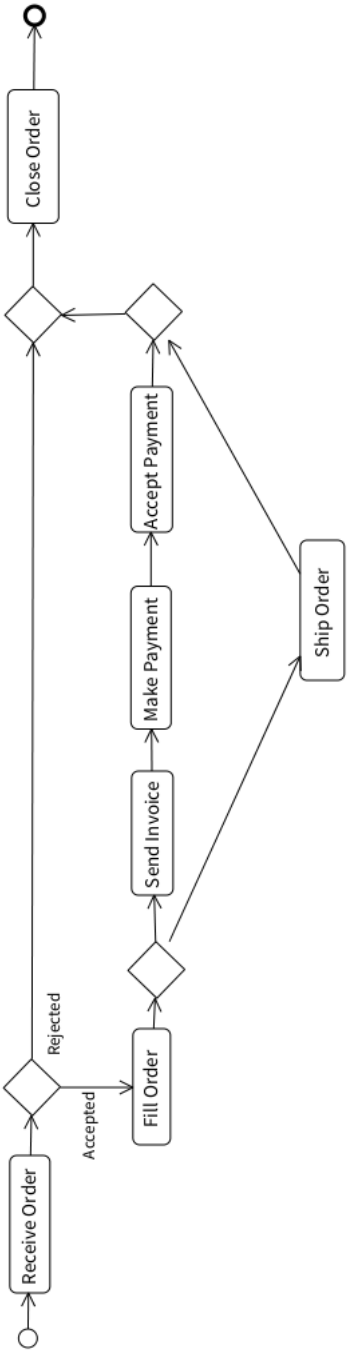
Největším problémem, se kterým jsem se setkal při vývoji, bylo hledání, jaký kód se volá. Kvůli tomu, že Pharo je dynamický jazyk, tak typy nejsou předem známy. Zkoumáním jednotlivých tříd jsem strávil hodně času.

Další problémy mi přinesl sám editor kódu. Jsem zvyklý na IDE, ve kterých

je rychlejší a inteligentnější našeptávaní. Pharo má pomalý a omezený editor, který je navíc case sensitive. Také pro mě bylo obtížné nepoužívat klávesové zkratky, na které jsem zvyklý. Po nějakém čase jsem ale problém odboural. Pharo má nestandardní metodu spuštěni kódu. To je ale na druhou stranu výhodou.

Bez ohledu na problémy se líbilo mi pracovat s jazykem Pharo a platformou OpenPonk. Platforma poskytuje skoro všechno, co je potřeba pro vývoj pluginu. Nevyzkoušel jsem si práci s GUI editoru, ale myslím si, že bude fungovat také dobře. Doufám, že výsledky mé práce pomohou někomu jinému při dalším rozvoji platformy OpenPonk.

Na konci této kapitoly ještě chci uvést obrázek 5.1, na kterém je stejný příklad jako ten, jež jsem použil na začátku této práce. Chybí tam některé typy gateway, které nebyly implementovány v rámci teto práce. Propojení elementů není tak hezké, protože se šipky neumí pěkně vyrenderovat na obrázku. Daný model byl otestován pomoci validátoru diagramu a je validní.



Obrázek 5.1: Výsledný diagram v OpenPonk

Kapitola 6

Možnosti budoucího vývoje

6.1 Bugy

Jako první krok dalšího vývoje vidím odstranění velkých bugů platformy.

6.2 Elementy

V teto práce jsem navrhnul malou množinu BPMN elementů. Tato množina umožňuje kreslit jenom jednoduché diagramy. Další rozšíření vidím v implementaci ostatních elementů, které by umožnily kreslit složitější modely.

6.3 Executable model

Generování „executable modelu“ je důležité pro EE. Díky tomu je možné provádět simulace běhu procesu. V této práci jsem se soustředil na „non-executable model“. Dalším krokem vývoje by mohlo být přidání funkcionality, která by generovala „executable model“.

6.4 Verifikace

Pokud budou vyvíjeny nové elementy, bude potřeba přidat nová pravidla pro nové elementy. Nebo dodělat další pro již existující elementy.

6.5 Simulace

Dalším krokem z pohledu budoucího vývoje vidím přidání možnosti simulace běhu procesu pro „executable modely“. Takový krok bude vyžadovat největší úpravy, ale přinese velkou výhodu oproti běžným BPMN editorům. Možná také bude potřeba upravit kód samé platformy OpenPonk.

6.6 Nové pluginy

Dobrou cestu vidím i ve vývoji nových pluginu pro OpenPonk. Zájem uživatelů o platformu je velmi důležitý. Díky dalším pluginum, které umožní používat nové typy diagramů, se může zvyšovat. Se zvyšujícím zájmem uživatelů stoupá i kvalita produktů.

Závěr

Jedním z cílů rešeršní částí bylo seznámení se s notaci BPMN a jejím uplatnění v enterprise engineeringu. Dalším cílem bylo navržení verifikačních pravidel pro BPMN diagramy. Hlavním cílem implementační částí prací byla implementace samotného pluginu pro práce s BPMN na platformě OpenPonk. Požadovanou funkcionalitou pluginu byla možnost verifikace BPMN diagramu.

Po analýze platformy OpenPonk a zjištěni pravidel implementaci dalších modulů, jsem vytvořil návrh svého pluginu. Vytvořeny plugin umožňuje kreslit jednoduché procesní diagramy. Další důležitou funkcionalitou je verifikace nakreslených diagramu.

Pro ověřeni správnosti implementace jsem vytvořil testy.

V budoucnu by bylo možné rozšířit plugin o ostatní prvky BPMN notace, které nebyly implementovány v rámci teto práce.

Literatura

[1] Hammer M., Champy J.: *Reengineering the corporation: A manifesto for business revolution. Revised, Updated edition.* HarperBusiness, 2006, ISBN 978-0060559533.

[2] Naplava Pavel, Pergl Robert: Empirical Study of Applying the DEMO Method for Improving BPMN Process Models in Academic Environment. 2015.

[3] Fedorov I.G.: *Modelirovanije biznes-procesov v notacii BPMN 2.0.* MESI, 2013, ISBN 978-5-7764-0772-7.

[4] Kolektiv autorů McKinsey & Company: Lean Russia: Sustaining economic growth through improved productivity [online]. apr 2009, [cit. 2018-04-23]. Dostupné z: `https://www.mckinsey.com/global-themes/employment-and-growth/lean-russia-sustaining-economic-growth`

[5] Donald H. Liles, Adrien R. Presley: ENTERPRISE MODELING WITHIN AN ENTERPRISE ENGINEERING FRAMEWORK [online]. dec 1996, [cit. 2018-04-23]. Dostupné z: `https://ieeexplore.ieee.org/document/873395`

[6] Dietz, J. L.: *Enterprise Ontology. Theory and Methodology.* Springer-Verlag Berlin Heidelberg, 2006, ISBN 978-3-540-33149-0.

[7] Palmer, N.: What is BPM?. In: *Bpm.com* [online]. mar 2014, [cit. 2018-04-23]. Dostupné z: `https://bpm.com/what-is-bpm`

[8] Nepal, M.: Business Process Management overview. In: *Workflow Management Software, Business Process Management (BPM) Software — KiSSFLOW* [online]. mar 2018, [cit. 2018-04-23]. Dostupné z: `https://kissflow.com/bpm/business-process-management-overview`

[9] Morphy T: What is BPM? — Business Process Management. In: *Stakeholder Mapping* [online]. [cit. 2018-04-23]. Dostupné z: https://www.stakeholdermap.com/bpm/bpm.html#design

[10] Martin: Business Process Management Life Cycle. In: *Cleverism* [online]. mar 2017, [cit. 2018-04-23]. Dostupné z: https://www.cleverism.com/business-process-management-life-cycle

[11] Martin: The Ultimate Guide to Business Process Management (BPM). In: *SweetProcess* [online]. mar 2017, [cit. 2018-04-23]. Dostupné z: https://www.sweetprocess.com/business-process-management

[12] Vercruysse, J.: The 5 basic elements of the BPM Life cycle. In: *Business processes for the future* [online]. dec 2017, [cit. 2018-04-23]. Dostupné z: https://www.effic.be/en/5-basic-elements-of-the-bpm-life-cycle

[13] Activevos Polančič team: Business Process Management System. In: *Informatica ActiveVOS BPMS* [online]. [cit. 2018-05-04]. Dostupné z: http://www.activevos.com/learn/business-process-management-system

[14] Kinzabulatov, R.: Chto takoje BPMS?. In: *Lutshije publikacii za sutki Habr* [online]. dec 2015, [cit. 2018-04-23]. Dostupné z: https://habrahabr.ru/company/trinion/blog/273025

[15] Object Management Group: ABOUT THE BUSINESS PROCESS MODEL AND NOTATION SPECIFICATION VERSION 2.0. In: *OMG — Object Management Group* [online]. jan 2011, [cit. 2018-04-23]. Dostupné z: https://www.omg.org/spec/BPMN/2.0/#specification-metadata

[16] Bonitasoft: BPMN 2.0. In: *Bonitasoft Documentation* [online]. [cit. 2018-05-02]. Dostupné z: https://documentation.bonitasoft.com/5x/bos-59/process-design/bpmn-20

[17] White, S. A.: Introduction to BPMN. dec 2017, [cit. 2018-04-23]. Dostupné z: http://omg.org/bpmn/Documents/Introduction_to_BPMN.pdf

[18] Hesse, M.: BPMN Tool Matrix. In: *BPMN Tool Matrix* [online]. aug 2017, [cit. 2018-04-23]. Dostupné z: https://bpmnmatrix.github.io/

[19] Object Management Group: Business Process Model and Notation (BPMN). In: *OMG — Object Management Group* [online]. jan 2011, [cit. 2018-04-23]. Dostupné z: https://www.omg.org/spec/BPMN/2.0/PDF

[20] Aagesen Gustav, Krogstie John: *BPMN 2.0 for Modeling Business Processes*. Berlin: Springer, 2015, ISBN 978-3-642-45099-0, str. 223.

[21] Kolektiv autorů: *On the Move to Meaningful Internet Systems: OTM 2012*. Berlin: Springer, 2013, ISBN 978-3-642-33606-5.

[22] Kolektiv autorů: *Software engineering research, management and applications*, kapitola Preserving Intentions in SOA Business Process Development. Berlin: Springer, 2008, ISBN 978-3-540-70561-1.

[23] Nishadha: Business Process Modeling Techniques with Examples. In: *Diagram Maker — Online Diagram Software — Creately* [online]. aug 2017, [cit. 2018-04-25]. Dostupné z: `https://creately.com/blog/diagrams/business-process-modeling-techniques`

[24] Polančič, G.: The Popularity Of BPMN Just Keeps Rising *Good e-Learning Blog* [online]. nov 2017, [cit. 2018-05-03]. Dostupné z: `http://blog.goodelearning.com/subject-areas/bpmn/popularity-bpmn-rising`

[25] Matthias Geiger, Simon Harrer, Jörg Lenhardb, Guido Wirtz: BPMN 2.0: The state of support and implementation [online]. jul 2016, [cit. 2018-04-25]. Dostupné z: `https://ac.els-cdn.com/S0167739X17300250/1-s2.0-S0167739X17300250-main.pdf?_tid=999b3004-2f91-404a-bacc-8688b6080aa9&acdnat=1525177253_d27264d666c5a0f8e5d6a27cff465bbf`

[26] Silver, B.: *BPMN Method and Style: with BPMN Implementer's Guide*. Cody-Cassidy Press, druhé vydání, 2011, ISBN 978-0982368114.

[27] Object Management Group: Business Process Modeling Notation (BPMN). jan 2009: str. 104, [cit. 2018-04-23]. Dostupné z: `https://www.omg.org/spec/BPMN/1.2/PDF`

[28] Blizničenko, J.: *Podpora simulace a vizualizace v nástroji DynaCASE*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství, 2015.

[29] Kolektiv autorů: OpenPonk. GUI. *OpenPonk software and business modeling platform* [online]. [cit. 2018-04-23]. Dostupné z: `https://openponk.github.io/dev/gui`

[30] Kolektiv autorů: OpenPonk verze od 9.10.2017 [zdrojovy kod softwaru]. [cit. 2018-04-23]. Dostupné z: `https://github.com/openponk/openponk`

[31] Kolektiv autorů: OpenPonk. Palette. *OpenPonk software and business modeling platform* [online]. [cit. 2018-04-23]. Dostupné z: `https://openponk.github.io/dev/palette`

[32] kolektiv autorů: OpenPonk. Navigator. *OpenPonk software and business modeling platform* [online]. [cit. 2018-04-23]. Dostupné z: `https://openponk.github.io/dev/navigator`

[33] Kolektiv autorů: OpenPonk. Extending menus. *OpenPonk software and business modeling platform* [online]. [cit. 2018-05-01]. Dostupné z: `https://openponk.github.io/dev/menus/`

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface (Grafické uživatelské rozhraní)

BPMN Business Process Model and Notation

BPM Business Process Management

BPMS Business Process Management Suite

MVC Model-View-Controller

EE Enterprise Engineering

IS Informační systém

OMG Object Management Group

ČVUT České vysoké učení technické v Praze

FSM Finite State Machine

BPMI Business Process Management Initiative

UI User interface (Uživatelské rozhraní)

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `impl` ..... adresář se spustitelnou formou implementace
  - `exe` ..... adresář se spustitelným OpenPonk s implementací
    - `linux` ..... adresář s OpenPonk pro Linux s implementací
    - `osx` ..... adresář s OpenPonk pro OS X s implementací
    - `win` ..... adresář s OpenPonk pro Windows s implementací
- `src` ..... zdroje textu práce a zdrojový kód pilginu
  - `text` ..... zdroje textu práce
    - `bp_anisimov.bib` .... použité informační zdroje ve formátu BibTeX
    - `bp_anisimov.tex` ..... text práce ve formátu LaTeX
    - `bp_anisimov.pdf` ..... text práce ve formátu PDF
  - `code` ..... zdrojový kód pilginu
    - `BPMN.st` ..... exportovaný balík BPMN z OpenPonk