ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Seliverstov** Jméno: **Aleksandr** Osobní číslo: **439589**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Aplikace pro ověřování dodržování standardů notace BPMN

Název bakalářské práce anglicky:

Application for verification of BPMN compliance

Pokyny pro vypracování:

Navažte na diplomovou práci J. Polana, který ověřil možnosti tvorby BPMN modelů a jejich následné testování prostřednictvím nástroje Camunda. V práci se zaměřte na dodržování standardů notace BPMN a nejčastější chyby modelů.
Postupujte následovně:
1) Definujte pojmy proces, procesní řízení.
2) Seznamte se s notací BPMN ve verzi 2.0.
3) Proveďte rešerši nejčastějších chyb tvorby modelů v notaci BPMN.
4) Navrhněte a implementujte aplikaci, která umožní uživateli vytvořit model v notaci BPMN, ověřit, zda odpovídá specifikaci notace a neobsahuje chyby, identifikované v předchozím bodu:
a. Model bude obsahovat obecné artefakty (aktivita, větvení, událost, bazén, plavecká dráha).
b. Na vyžádání uživatele proběhne kontrola správnosti modelu.
c. V případě nalezení chyby bude na ni uživatel upozorněn, v modelu bude naznačeno, kde se chyba nachází a uživatel dostane informace, které mu pomohou chybu odstranit.
5) Funkčnost aplikace ověřte formou uživatelského testování na vybrané skupině uživatelů.

Seznam doporučené literatury:

[1] Jan Polan, Virtuální příprava procesních analytiků, diplomová práce, ČVUT FEL, 2019
[2] BPMN Method and Style, 2nd Edition, with BPMN Implementer's Guide- Silver, B., Cody-Cassidy Press, 2011
[3] Modelování podnikových procesů (online), - Klimeš, C., Ostrava 2014, dostupné z http://www1.osu.cz/~zacek/mopop/mopop.pdf

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Náplava, Ph.D., katedra ekonomiky, manažerství a humanitních věd FEL

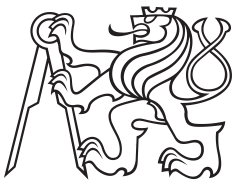
Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2021**

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Aplikace pro ověřování dodržování standardů notace BPMN

Aleksandr Seliverstov

Vedoucí práce: Ing. Pavel Náplava, Ph.D.
Srpen 2020

Poděkování

Chtěl bych poděkovat co nejvíc svému vedoucímu práce, doktoru Náplavovi, za jeho vstřícnost a cenné rady během mého vypracování této práce. Další poděkování bych směřoval své rodině a přátelům–včetně přátelů za oceánem–za jejich podporu. Také bych chtěl poděkovat testerům této aplikace za jejich čas a ochotu mně s testováním pomoct a mému korektorovi, který pomohl mně s opravou češtiny.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem č.1/2009 o dodržování etických principů při přípravěvysokoškolských závěrečných prací.

V Praze, 14. srpna 2020

Abstrakt

Tato bakalářská práce se zabývá procesním řízením, modelováním procesů v notaci BPMN a jejich následnou analýzou. V první části je definována teorie související s procesem a procesním řízením. V části následné je popsán základní úvod do problematiky modelování procesů, dál v části navazující uveden popis notaci BPMN a jsou krátce zmíněny zodpovědnosti procesního analytika. Součástí práce je také rozbor nejčastějších chyb procesních analytiků, které v další části práce je využito pro návrh a implementace aplikace, schopnou tyto chyby detekovat a upozornit uživatele na nich. Poslední část práce obsahuje výsledky uživatelského testování této aplikace.

Klíčová slova: Procesní řízení, modelování procesů, BPMN, časté chyby, validace procesů

Abstract

This bachelor thesis is dedicated to process management, process modeling in BPMN notation and the following analysis of said models. In the first part of the thesis is the theory covering the topics of process and process management. The next part introduces process modeling and the part following after covers BPMN modeling notation in further details, the responsibilities of process analyst are also mentioned. Another part of the thesis is a breakdown of commonly made mistakes during the process modeling. This data is used in the later part of the thesis to define and implement an application capable of detecting and pointing out said mistakes. The last part of the thesis consists of feedback received during the testing phase of the application.

Keywords: Process management, process modeling, BPMN, common mistakes, process validation

Obsah

Obrázky

Tabulky

Úvod

Tato práce se zabývá problematikou procesního řízení, modelováním procesů v notaci BPMN s dodržováním oficiálních standardů a zlepšením dovedností procesních analytiků.

Cíle práce

Cíle práce jsou rozděleny do dvou podčástí: první části je těsné seznámení s procesním řízením a notací BPMN. Po seznámení s notací je potřeba provést rešerši zabývající se analýzou nejčastějších chyb, ke kterým se dochází při tvorbě modelů v této notaci v praxi. Druhou částí je návrh a implementace aplikace na základě výsledků z částí první. Výsledný software by měl splňovat obecné požadavky uvedené v zadání práce.

Motivace práce

Motivací pro tuto práci bylo vytvoření nástroje, který by šlo využít ve výuce jako pomůcku pro studenty, kteři tvoří procesní modely a pomocí této aplikace by mohli předejít nejčastějším chybám, které dělají procesní analytici i v praxi. Jako osobní motivaci bych uvedl zájem o téma a snahu dosáhnout definované cíle, včetně vytvoření aplikace, která by mohla usnadnit studentům a vyučujícím výuku, především však studium procesního řízení.

Struktura práce

Práce je rozdělena do dvou částí: teoretické a praktické. Prvních čtyř kapitol je věnováno první části práce. Další čtyři kapitoly jsou věnovány druhé části - samotné aplikaci, která vznikla jako výsledek této práce. V první kapitole je popsán proces a s ním spojené pojmy a vlastnosti, bez kterých by nešlo pokračovat do procesního řízení. Následně v této kapitole popisuji procesní řízení, jeho principy a základní aspekty s ním spojené. Druhá kapitola je věnována modelovaní procesů a slouží jako úvod do kapitoly číslo tři, která podrobně popisuje notaci BPMN, se kterou v rámci této práce pracuji. V čtvrté kapitole uvádím popis a objasnění nejčastějších chyb, ke kterým se dochází během modelování procesů v notací BPMN. V kapitole číslo pět definuji požadavky na aplikaci, které jsou navázány na výsledky první částí práce. Šestá kapitola obsahuje v sobě popis logiky chování aplikace. V sedmé kapitole uvádím popis práce s aplikací a popis implementace včetně využitých technologií. V osmé kapitole popisuji výsledky uživatelského testování a krátce popisuji možné návrhy s ohledem na tyto výsledky a diskutuji budoucnost aplikace. Poslední kapitola je věnována shrnutí stavu aplikace a vyhodnocení její efektivity.

Kapitola 1

Proces a procesní řízení

V této kapitole se zabývám procesem a s tímto pojmem spojenými vlastnostmi. Na začátku definuji pojem procesu a pokračuji podrobným popisem rozdělení a vlastností procesů. Na konci uvádím náhled do životního cyklu procesu. Dál pokračuji v definování procesního řízení a následně popisuji jeho problematiku. Rozebírám jeho základní principy a na konci uvádím výhody a nevýhody procesního řízení.

1.1 Proces

1.1.1 Definice procesu

Podle definice ČSN EN ISO 9000:2000 [10] proces je chápán jako:

> *„soubor vzájemně souvisejících nebo vzájemně působících činností, který přeměňuje vstupy na výstupy“*

Jinak řečeno, během každého procesu jedná se o výměnu vstupních zdrojů (jako hmotných, tak i nehmotných) za výstupy. Ke vstupním zdrojům mohou patřit peníze, materiál (hmotné zdroje), know-how, license (nehmotné). Jak již bylo zmíněno, tyto činnosti přeměňují zdroje na výstup, například služby nebo výrobky. Tyto výstupy jsou pak zhodnocené zákazníkem procesu.

Proces jako takový má přesnou trajektorii, má cíl, ke kterému se blíží. Jinými slovy každý proces musí se začít v určitém bodě, projít nějakým dalším bodem a v určitém bodě dospět ke konci. Lze tedy říct, že proces

je skupina navazujících činností, které mění vstupy za výstupy takovým způsobem, aby splňoval požadavky zákazníka.

1.1.2 Rozdělení procesů

Existuje řada klasifikací business procesů a neexistuje jedna "nejsprávnější". Vždy se jedná o tom, která je nejvhodnější pro využití uvnitř organizace.

Podle Šmidy [11] je dělení na hlavní, řídící a podpůrné nejobvyklejší způsob klasifikace procesů. Toto dělení je podle jeho názoru jednoduché, napovídá a poskytuje důležité informace o procesu. Na základě toho v této práce bude využita přesně tato klasifikace rozdělení procesů.

- **Hlavní** jsou procesy, které generují zisk a v zásade jsou účelem existence organizace.
- **Řídící** jsou procesy, představující aktivity společnosti nutné pro její chod. I když sami o sobě nepřinášejí zisk, jsou důležitou součásti řízení společnosti.
- **Podpůrný**. Tento typ procesu svou činností podporuje předchozí dva. Jsou však důležite, protože bez nich by hlavní procesy nemohli v pořádku fungovat.

Pro lepší názornost rozdíly mezi typy procesů jsou v následující tabulce[12]:

	Hlavní	**Řídící**	**Podpůrný**
Přidává proces hodnotu?	Ano	Ne	Může
Probíhá napříč společnosti?	Ano	Může	Ne
Má externího zákazníka?	Ano	Ne	Ne
Generuje tržby?	Ano	Ne	Ne

Tabulka 1.1: Rozdíly mezi typy procesů

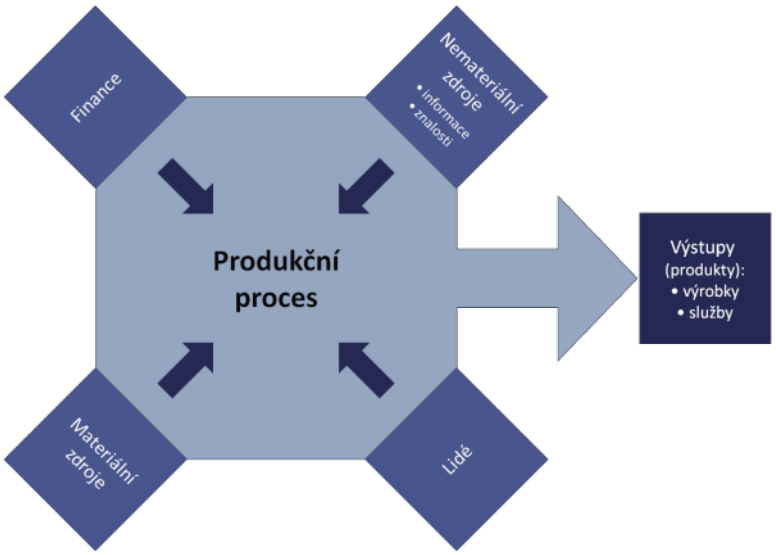
1.1.3 Vlastnosti procesů

Protože typů procesů jsou spousta, tato podkapitola bude věnována vlastnostem podnikových procesů na základě jejich rozdělení na hlavní, řídící a podpůrné. Jinými slovy, tyto vlastnosti obecně platí pro procesy uvnitř organizací, které modelují svoje podnikové procesy na základě tohoto rozdělení.

Mezi vlastností takových procesů patří:

- Nutnost mít dodavatele či vstup na začatku a zákazníka na konci
- Dekompozice (Lze proces rozložit na podprocesy a aktivity)
- Probíhání sekvenčně a opakovaně
- Každý proces má nutně vlastníka

Někteři definují produkční proces, který prochází napříč celou organizací, jako základní kostru procesů v organizaci.[13] Na následujícím obrázku 1.1 je znázorněno, jak schematicky vypadá takový typ procesu.



Obrázek 1.1: Schéma produkčního procesu. [1]

Je dobře vidět, že toto schéma vyplývá z již zmíněných vlastností procesů a popisuje definici procesu jako takového: provádí se změna různých vstupů za výstupy. Přesto je potřeba vydefinovat účastníky a objekty, které procesu zúčastní.

- **Vstup procesu** je objekt nebo jeho stav před působením zkoumaného procesu.
- **Výstup procesu** je objekt nebo jeho nový stav po proběhnutí zkoumaného procesu.
- **Činnosti** jsou aktivity výkonáné v rámci procesu v předem definovaném sledu. Příkladem činností mohou být
  - Zahájení aktivity (podle plánu, po osobním požadavku, po změně situace)
  - Rozhodnutí (například na základě kvalifikovaného odhadu nebo hlasování)

  - Změny (působí na objekt nebo okolí)
  - Ukončení (musí po ukončení být nějaky výstup – hmotný či nehmotný objekt nebo stav objektu po ukončení působení aktivity)

- **Zdroje** procesu jsou všechny objekty potřebne pro výkon aktivit a činností během procesu.

- **Vykonovatel (dodavatel)** procesu je osoba zodpovědná za provedení aktivit spojených s procesem. V případě, kdy dodavatelem je proces na který navazuje daný proces, nejedná se o dodavatele jako osobu či firmu, ale o výstup toho procesu.

- **Vlastník procesu** je právě jedna osoba zodpovědná za nastavení procesu, tj. za jednotlivé aktivity, provádění procesu, dodržování postupů s cílem uspokojení požadavků zákazníka. Vlastník obecně zajišťuje řízení a zlepšování projektu, např. má právo k provádění změn v procesu, pokud je to žádoucí.

- **Zákazník** je osoba nebo subjekt, pro které je určen výstup procesu. Obecně se očekává pravidelná komunikace mezi dodavatelem a zákazníkem a včasné informování o spokojenosti s dodávanými výstupy.

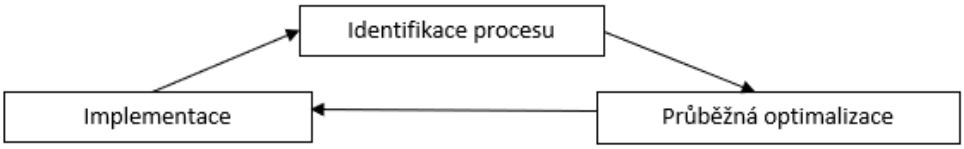
- **Cíle procesu** jsou jasně definované a očekávané výstupy po provádění procesu a dávají vědět, k čemu ten proces směřuje. Navíc definují požadavky, kterých proces by měl dosáhnout pro uspokojení minimálních potřeb zákazníka. V cílech procesu se taky může definovat časové rozmezí pro vykonávání činností.

- Každý proces má související **rizika** při provádění. **Rizika procesu** jsou seznamem možných výskytů během procesu a této výskyty budou záporně ovlivňovat výstupy procesu. Obecně pro každé riziko existuje mitigační a krizový plán.

1.1.4 Životní cyklus procesu

Identifikace procesu a jeho následné řízení ve společnostech je jen první krok v životním cyklu procesu a úspěšném řízení organizace. S procesem je vždy potřeba neustále pracovat, zlepšovat, optimalizovat a hledat způsoby zvýšení příjmů pro organizace. Navíc, pomocí stálé kontroly a optimalizace procesu, organizace snižuje možnost velkých ztrát, protože nemůže dojít k nevčasnému či pozdnímu zjištění jeho neefektivity. Po identifikaci tohoto problému organizace může bud' zahájit potřebné kroky pro změnu procesu, nebo ten proces zrušit.

Celý životní cyklus procesu lze znázornit následujícím grafem1.2 [14].



Obrázek 1.2: Životní cyklus procesu

Analýza procesů se by měla provádět v určitých časových intervalech, aby nedošlo ke zvýšení nákladů v případě pozdního zjištění neefektivity nebo redundance procesu. [14]

1.2 Procesní řízení

1.2.1 Definice procesního řízení

Pojmem procesního řízení se obecně myslí řízení organizace takovým způsobem, který zdůrazňuje opakované procesy a jejich průběh napříč celou organizací. Podle ITIL/ITSM [15] procesní řízení je soubor činností týkajících se plánování a sledování výkonnosti především realizačních firemních procesů.

1.2.2 Základní principy procesního řízení

Mezi principy procesního řízení obvykle se uvádí následující:[12]

- **Integrace a specifikace práci**. Všechny činnosti se spojují do jednoho procesu tak, aby nad ním pracoval jeden tým a výstup by měl co nejvyšší hodnotu pro zákazníka.
- **Delinearizace činností**. Činnosti by měly se vykonávat v určitém sledu a o tom, jak jsou činnosti na sebe navázané, rozhoduje pracovní tým.
- **Nejvýhodnější místo realizace činností**.

  > *„Práce se vykonávají tam, kde je to nejvýhodnější, bez ohledu na organizační hranice uvnitř podniku i mimo podnik."[16]*

- **Odpovědnost za proces**. Každý proces má svého vlastníka, který je zodpovědný za řízení tohoto procesu a plnění cílů.
- **Procesně zaměřená motivace**. Výsledek procesu je přímo závislý na motivaci pracovníků.

- **Podpora flexibility procesů**. Výstupy procesů by měli co nejvíce odpovídat požadavkům jiných trhů nebo zákazníků. Jinými slovy, míra flexibility procesů by měla dovolit vyprodukování produktů dle individuálních potřeb zákazníků.
- **Podpora principu 3S**. Princip samořízení, samokontroly, samoorganizaci by měl členům týmu zvýšit jejich znalosti a odpovědnosti za vlastní práci.
- **Znalostní a informační bezbariérovost**. Tento princip je založen na odstranění znalostních a informačních bariér vytvořením databáze znalostí, údajů a informačních zdrojů.

1.2.3 Výhody procesního řízení

Jako hlavní výhody a přínosy procesního řízení jsou v literatuře označeny: [17]

- **Zprůhlednění organizace**. Každá organizace má své definované pracovní postupy a chování. Popsáním a standardizací těchto postupů usnadníme řízení procesů a umožníme lépe definování vztahů mimo firmu (v případě spolupráci s dalšími firmy).
- **Snadná a rychlá optimalizace procesů**. Pokud organizace má definované a dokumentované průběhy procesů, je velmi snadné a rychlé (díky dostatečnému množství informací) tyto procesy optimalizovat tak, aby reflektovaly nové požadavky trhu.
- **Definovaná odpovědnost**. Každý proces má jednoznačně definovanou zodpovědnou osobu, která za ten proces nese odpovědnost. Takovým způsobem je zodpovědnost dobře vystopovatelná a umožňuje se využití kompetenčního modelu na všech úrovních organizace.
- **Uchovávání know-how**. Protože know-how je jednou z největších hodnot společností, její ukládání v procesech nebo jejich popisech je velkou výhodou pro organizace.
- **Podpora v informačních technologiích**. Optimalizované průběhy procesů mohou být podpořeny prostřednictvím informačních systémů. Tím je umožněno efektivní využití procesů. Příkladem využití procesu lze uvést situaci, kdy proces je namodelován v informačním systému. V tomto případě je proces kontrolován přímo tímto informačním systémem.
- **Podpora požadavků ISO norem**. Hodně společností usiluje o dosažení určitého stupně kvality a dodržením standardu certifikátu ISO, je garantováno, že mají popsané a implementované procesy uvnitř organizace.

1.2.4 Nevýhody procesního řízení

Určitě jsou i situace, kdy využití procesního řízení má své nevýhody. Mezi nimi patří: [18]

- **Obtížný přechod na nový způsob řízení**. Tato nevýhoda je často spojena s přechodem z funkčního řízení na procesní. Nejedná se o triviální krok: je třeba překonat funkční způsob myšlení, změnit kulturu podniku, určit řadu technologických změn. Dokonce se velmi často stává, že přechod na nový způsob řízení není dokončen a změna se nekoná.
- **Neochota zaměstnanců popisovat a překonávat know-how**. Většina zaměstnanců není ochotná popisovat a překonávat know-how z důvodu ztracení své výhody pro organizaci a následně jejich nahraditelnosti.
- **Méně častý výskyt pozic Procesní analýtik (Business Analyst) nebo Procesní návrhář (Process Designer)**. U procesního řízení je podkladem optimalizace a modelování procesu. Tímto většinou by se měli zabývat zaměstnance na pozici procesního analytika nebo procesního návrháře.

1.3 Závěr

V první kapitole jsem popsal pojem proces, jeho rozdělení a vlastnosti. Pro lepší pochopení jsem vydefinoval životní cyklus procesu, aby bylo jasně, jaké kroky proces během své existence prochází. Tato kapitola by měla sloužit jako vstupní seznámení s procesem pro snadnější přechod do další její části, ve které detailněji rozebírám problematiku procesního řízení. Uvedl jsem v rámci tohoto popisu základní principy procesního řízení a neopomněl jsem probrat výhody a nevýhody, které procesní řízení se sebou přináší. V další kapitole se již budu věnovat modelování procesů.

Kapitola 2

Modelování procesů

Protože součástí procesního řízení je modelování procesů, tato kapitola slouží jako seznámení s její problematikou. První sekce popisuje úvod do modelování procesů. Další částí je krátký popis přínosů procesního modelování a účelů procesního modelování v praxi. Na konci rozebírám typy modelování a popisuji fáze modelování v praxi.

2.1 Úvod do modelování procesů

V úvodu bych uvedl definici a účel modelování procesů. Modelování procesu lze definovat, jako například v následujícím odstavce:

> *"Modely podnikových procesů představují přehled procesů probíhající ve společnosti. Pomocí namodelovaných procesů získáváme jednodušeji informace a celkový přehled o procesech, které v podniku probíhají. Je to prostředek, pomocí kterého můžeme ve společnosti popsat a porozumět svému vnitřnímu uspořádání a chování."[19]*

Tímto se myslí, že pomocí modelování procesů znázorňujeme stávající procesy v podniku a toto nám usnadňuje jejich analýzu. V dalším odstavce je krátce popsáno k čemu modelování procesů se využívá a dána další možná definice pojmu.

> *“Modelování procesů je široce využíváno uvnitř organizací jako způsob zvýšení povědomí a znalostí týkajících se podnikových procesů*

a pro dekonstrukce organizační komplikovanosti. Je to přístup popisující provoz firmy a obecně zasahuje do grafického znázornění činností, stavů a toků zpráv, které spolu tvoří podnikový proces." [20]

Obvykle se modelují procesy probíhající uvnitř jednotlivé firmy, tj. modely nezasahují do procesů mimo firmu. Avšak v závislosti na účelu modelu namodelovaný podnikový proces může obsahovat i procesy mimo organizaci. Protože každý model má svůj specifický účel, dva modely popisující stejný proces se mohou mírně lišit v tom, jak vypadají. Je to spojeno s tím, že modely můžou v sobě mít odlišné míry detailu a míra detailu obvykle je stanovena v závislosti od toho, pro koho ten model byl vytvořen. Obecně modelování procesů je využíváno pro zlepšení kvality a efektivity firmy. Pro grafické znázornění procesů je potřeba vybrat a využít vizuální nástroj. Pro to můžeme využít obecné modelovací notaci určené pro modelování systému (UML, DFD, Petriho sítě), nebo notaci určené pro modelování procesů jako BPMN.

2.2 Přínosy procesního modelování

K přínosům procesního modelování pro firmu lze uvést: [21]

- **Zlepšení efektivity**. Hlavní účel modelování je zlepšení již existujících procesů. Pomocí modelování procesů snadno znázorníme průběh procesů a budeme moct navrhnout způsoby zefektivnění, což bude vést k lepším výstupům a přínosům.
- **Agilnost procesů**. Je-li modelování procesů ve firmě prováděno pravidelně, způsobuje to rozvoj kultury inovace a zlepšení ve firmě. Pokud se často provádí reiterace ohledně běhu procesů, bude se firma moci snadno přizpůsobit technologickým změnám na trhu.
- **Průhlednost**. Všichni uvnitř firmy budou vědět, jak procesy probíhají. Toto povede k jasnějšímu rozdělení zodpovědností mezi zaměstnance.

Z toho lze usoudit, že modelování procesů je v zásadě nástrojem pro rozvíjení a optimalizaci běhu firmy s účelem zvýšení efektivity podnikání a přínosů.

2.3 Typy a fáze modelování

Přístupy ke grafickému znázornění procesu mohou být rozděleny do dvou částí: **funkcionální** a **činnostní** (řízený činnostmi). Funkcionální přístup reprezentuje sadu vzájemně propojených funkcí s povinným uvedením vstupních, výstupních a řídících toku. K notacím, které náleží do funkcionálního přístupu patří: [22]

- SADT (Structured Analysis and Design Technique) a IDEFO (Integration Definition for Function Modeling)
- Modelování podnikových procesu a bloků 1DEF3
- Křížově funkcionální vývojový diagram (Cross Functional Flowchart)
- Metoda Ericsson-Penker, založena na rozšířené variantě jazyka UML (Unified Modeling Language)

Činnostní přístup reprezentuje model jako systém procesů. Základem tohoto přístupu je řízení podnikových procesu prostřednictvím předání informace o změnách probíhajících uvnitř procesu nebo v jiných procesech. K notacím činnostního přístupu patří:[22]

- EPC (Event-driven Process Chain)
- BPMN (Business Process Model and Notation)

Modelování procesu se obecně, ale ne vždy, dělí na tři fáze: sběr informace, modelování stavu AS-IS a modelování cílového stavu TO-BE.

- **Sběr prvotní informace zahrnuje identifikaci všech systémů a v nich probíhajících procesů**. V tomto kroku se provádí analýza dokumentace, podnikových procesů, aktérů. Výstupem těchto aktivit je dokument, na základě kterého se provádí modelování procesů.
- **Modelování stavu AS-IS**. Tento krok v sobě zahrnuje výběr přístupu k popisu podnikových procesů, a vytváření původního modelu odpovídající současnému stavu ve firmě. Vytvořený modely se pak analyzuje a případně se mění až do konečné verze. Tato konečná verze modelu je výstupem tohoto kroku.

- **Modelování stavu TO-BE**. Na základě analýzu z předchozího kroku se provádí návrh a následné nasazení modelu v stavu TO-BE–to, jak by měl model vypadat. Na základě výsledků nasazení se provádí další analýza a zlepšení modelu. Je tady důležité zmínit, že ne vždy se modeluje budoucí stav to-be. Například, kdyby podník po modelování procesu a jeho následně analýzě byl spokojen se stávájícím stavem, tak by v tomto případě budoucí stav se nemodeloval. Však z životního cyklu víme, že proces bychom vždy chtěli cílit optimalizovat. V tomto případě modelování stavu TO-BE by dávalo smysl.

2.4 Závěr

V této kapitole jsem popsal oblast modelování procesů. Začátek kapitoly obsahuje hlavní účely a přínosy procesního modelování pro firmy a slouží pro seznámení čtenáře s důležitostí procesního modelování v podniku. V další části jsem popsal přístupy (či tipy) modelování a uvedl jsem příklady notací, které se využívají v praxi pro každý z těchto typu. Na konci okrajově jsem popsal fáze modelování procesů, přes které prochází skoro každý model vytvořený procesním analytikem.

Kapitola 3

Modelovací notace BPMN 2.0

V této kapitole se zabývám modelovací notaci BPMN verze 2.0, která je jednou z nejvyužívanějších modelovacích notací ve světě. Na začátku definuji BPMN a jí cíle. Následně pokračuji popisem historie vzniku a vývoje této notace. Dále rozebírám a ilustruji základní prvky notace, včetně popisu jejich využití a významu. V další části ukazuji různé typy modelů, a pro zajímavost provádím porovnání BPMN s další populární a široce využívanou notací - UML. V závěrečné části kapitoly krátce popisuji profesi procesního analytika a jako shrnutí celé kapitoly ukazuji příklad namodelovaného procesu v notaci BPMN 2.0.

3.1 Definice BPMN

Předtím, než vydefinuji notaci BPMN, uvedu pojem "notace". Obecně řečeno, "notace"je systém podmíněných označení, akceptovaný v kontextu nějaké znalostní oblasti. Tento systém do sobě zasahuje věci jako abeceda či srozumitelné pojmenování a definice využitých značek, včetně pravidel jejich využití.

Business Process and Model Notation (BPMN) je standardizovanou notací poskytnutou Object Management Group (OMG). Hlavním cílem BPMN bylo vytváření notace, která by byla jednoduše pochopena všemi skupinami business uživatelů: od analytiků, které vytvářejí koncepty procesů do vývojářů, které se zabývají implementací technologií vykonávající této procesy. Jinými slovy, BPMN tvoří standardizovaný můstek, spojující návrh business procesů s jejich implementací. [11]

3.2 Historie BPMN

BPMN verze 1.0 byl vyvinut konsorciem BPMI (Business Process Management Initiative) v květnu roku 2004. Hlavním cílem bylo vytvoření notace čitelnou pro všechny účastníci business procesů. V roku 2005 došlo ke sloučení konsorcia OMG (Object Management Group) a BPMI a od této doby je notace BPMN spravována a rozvíjena touto skupinou.

Po tomto sloučení pod vedením OMG byla vypracována verze BPMN, která byla v únoru roku 2006 přijatá jako standardní. V červnu roku 2007 byla dokončena verze 1.1. Z této verze dál byla vyrobena verze 1.2, která byla vydána v roku 2008.

V lednu roku 2011 došlo k vývoji verze 2.0, která měla za hlavní úkol sjednotit BPMN a BPDM (Business Process Definition Metamodel) a došlo k rozšíření názvu této notací z Business Process Modelling Notation na Business Process Model and Notation. Přínosem nové verze by mělo být vytvoření jednotného jazyku s metamodelem, grafickou notací a výměnným formátem.

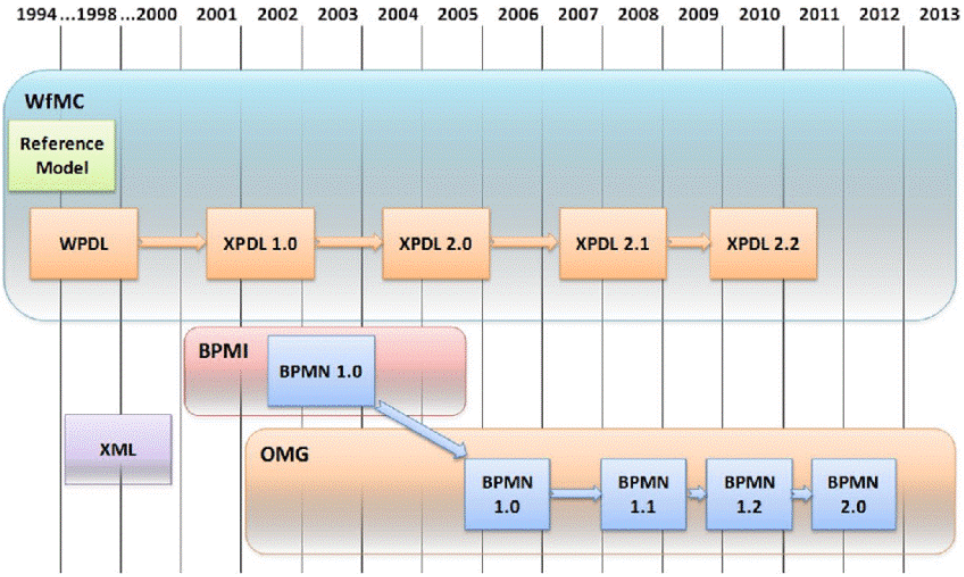
K přínosům nové verze také patří:

- Dořešení problémů odložených při dokončování BPMN 1.1 a BPDM 1.0
- Schopnost výměny business proces modelů a jejich diagramů mezi modelovacími nástroji se zachováním integrity
- Změny notaci potřebné k sjednocení BPMN a BPDM do jednoho konzistentního jazyka
- Podpora zobrazení a výměna odlišných pohledů na business process, co umožňuje vlastníku procesu se zaměřit na slabé strany
- Poskytnutí XML schématu sloužícího pro transformaci modelů

V době psaní této práce standardem je BPMN 2.0.2 vyvíjena v lednu roku 2014. K jejím přínosům patří:

- Řešení známých nekonzistencí a nejasností předchozí verze
- Definování choreografického modelu
- Formalizace provádění sémantiky pro všechny prvky
- Rozšíření modelů definujících lidské interakci

Grafické znázornění historie BPMN je uvedeno na následujícím obrázku 3.1: [23]



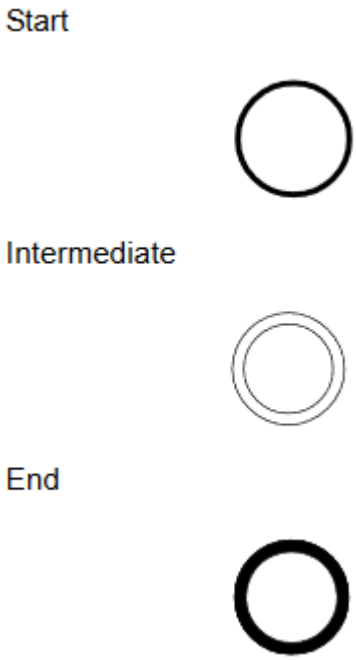
Obrázek 3.1: Časová osa vývoje BPMN (není na obrázku aktuálnější standard BPMN 2.0.2)

3.3 Grafické prvky BPMN

V této podkapitole popíšu nejčastěji se vyskytující prvky v notaci BPMN a jejich význam, včetně jejích grafického znázornění.

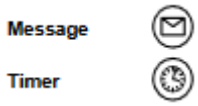
- Událost

  Událostí je něco, co se provádí během procesu. Tyto události mají vliv na model a mají příčinu (trigger) nebo dopad (result). Události jsou zobrazené jako kolečka a případně mají uvnitř další znaky pro lepší rozlišení příčin a dopadů. Obecně jsou tři typy událostí: Počáteční (Start), Prostřední (Intermediate), Konečná (End).



Obrázek 3.2: BPMN Event [2]

Události jsou mnoha druhů, ale nejčastěji používané jsou zpráva (message) a časovač (timer). Jsou zobrazené s obálkou a ciferníkem. Message obecně značí příjem nebo odeslání zprávy, v závislosti od směru toku. Timer znázorňuje čekání na časově založenou podmínku, po obdržení které se spouští činnost zachycení (Catch Event anglicky).



Obrázek 3.3: BPMN Event types [2]

- Aktivita

  Aktivity jsou základní prvky procesů. Aktivity mohou být jak atomické, tak i neatomické. Jsou zobrazené jako obdélníky.

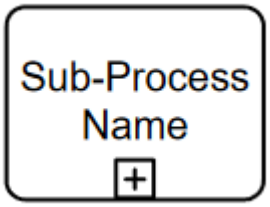
Obrázek 3.4: BPMN Activity [2]

- Úkol (Task)

  Úkol je atomickou aktivitou a využívá se ve případě, kdy aktivita nemůže být dekomponována na víc aktivit pro dosažení vyšší úrovně detailů.

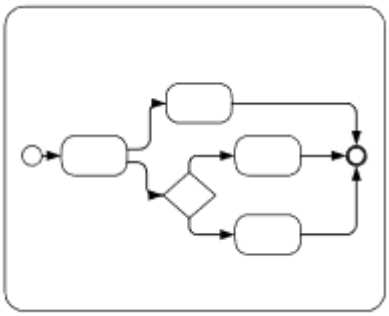
- Podproces (Subprocess)

  V případě, že aktivita je složena z několika větších nebo komplikovanějších procesů, využívá se podprocess. Pokud se jedná o zatahnutý (collapsed) podproces, ten je znázorněn stejně jako aktivita s rozdílem, že v dolní části se přidává čtverec se znakem plus (+) uvnitř, který je indikátorem další úrovně detailů.



Obrázek 3.5: BPMN SubProcess [2]

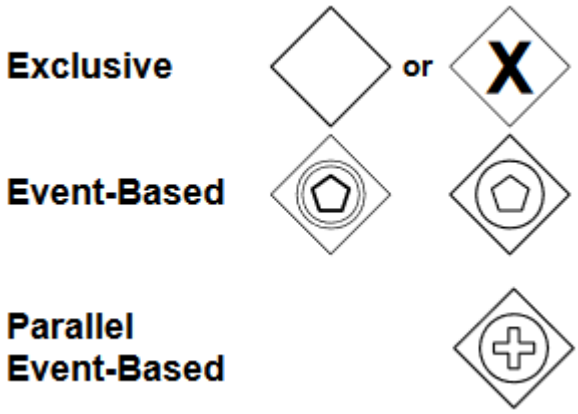
Pokud se jedná o roztáhnutý (expanded) podproces, detaily jsou vidět uvnitř hranic tohoto procesu. Důležité je, že sekvenční tok uvnitř podprocesu nesmí přejít hranice obdélníku.



Obrázek 3.6: BPMN SubProcess Expanded [2]

- Gateway

  Gateway (brána) se využívá pro označení větvení (branching), rozcestí (forking) a spojování (merging) sekvenčních toků během procesu. Tento prvek je znázorněn jako kosočtverec s případným symbolem uvnitř, který určuje typ brány.



Obrázek 3.7: BPMN Gateways [2]

- Sekvenční toky (Sequence Flows)

  Sekvenční tok ukazuje, v jakém pořadí budou prováděny aktivity během procesu. Je obecně znázorněn jako šipka. Další symboly určují typ toku.

Obrázek 3.8: BPMN Sequence Flow [2]

Závislý tok (Conditional flow) je typ sekvenčního toku kterým v procesu tok projde jen když byla splněna odpovídající tomuto toku podmínka. Je zobrazen jako šipka s kosočtvercem na začátku. Toto označení se využívá jen v případě, kdy tok vychází z aktivity, nikoliv z brány (gateway).

Obrázek 3.9: BPMN Conditional Flow [2]

Defaultní tok (Default flow) je využit jen v případě, kdy žádným z dalších toků odchozích z činnosti tok procesu neprošel. Označuje se jako šípka s lomítkem na začátku.

Obrázek 3.10: BPMN Default Flow [2]

3.4 Typy BPMN modelů

Podle aktuální specifikací BPMN 2.0.2 existuje tří dílčí typy BPMN modelů: [2]

- Procesy
  - Soukromé procesy (Public processes)
    - Spustitelné
    - Nespustitelné
  - Veřejné procesy (Private processes)
- Choreografie
- Kolaborace (Collaboration) [1]

Model procesu

Procesní model popisuje sekvence nebo tok aktivit v organizaci. V BPMN procesní diagram je definován jako graf, obsahující elementy jako aktivity, činnosti, brány a sekvenční toky mezi nimi. Procesy můžou být definované jak na úrovní celé organizace, tak i na úrovní jednoho zaměstnance.

V BPMN je využit pojem procesní model jen v případě kdy se jedná o sadu jednoduchých prvků. V případě modelování interakcí mezi procesy je využit pojem Kolaborace nebo Choreografie.

Soukromé procesy jsou obecně procesy uvnitř organizace. Podle specifikaci se jim říká workflow nebo BPM proces. V případě web služeb je také často využit pojem orchestrace služeb. Na následujícím obrázku 3.11 je znázorněn jednoduchý příklad soukromého procesu.

Obrázek 3.11: Příklad soukromého procesu uvnitř organizace [3]

Soukromé procesy jsou dvou typů: **spustitelné** (executable) a **nespustitelné** (non-executable). Spustitelným procesem je proces, který byl modelován

[1] V rámci této práce bude anglický pojem Collaboration přeložen jako kolaborace.

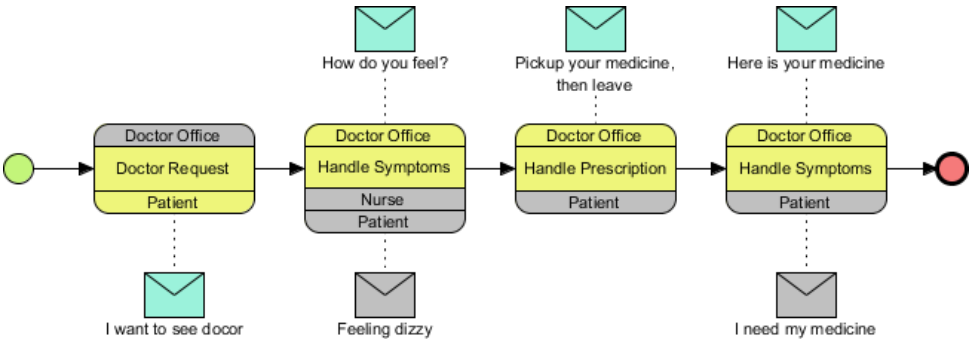
za účelem spuštění, tj. je v něm definovány vše potřebné náležitosti pro spuštění. Však během vývojářského cyklu procesu budou fáze, během kterých proces nebude mít dostatečnou míru detailu, aby byl spustitelným.

Nespustitelné procesy jsou procesy, které byly modelovány s účelem dokumentování chování procesu. Tudíž informace potřebná pro spuštění procesu, jako například formální podmínky Expressions (výraz) obecně nejsou součástí nespustitelných procesů.K příkladu nespustitelného procesu lze odnést model, kde nejsou popsány podmínky formálních výrazů (formal condition expressions anglicky) v modelu.

Choreografie

Choreografie je typ procesního modelu, který se liší od standardního modelu BPMN procesu svým cílem a chováním. Na rozdíl od procesního modelu, který definuje proces uvnitř organizace, choreografický model definuje a formalizuje způsob koordinace a komunikace mezi business účastnící. Business účastník je vlastně osoba nebo firma, se kterou v diagramu choreografie modelujeme průběh komunikace. Zaměření tohoto typu procesního diagramu neleží v definování komunikaci uvnitř každého z účastníků, ale ve výměně informace mezi těmito účastníky. Navíc choreografie nemá žádný centrální ovladač nebo pozorovatele procesu.

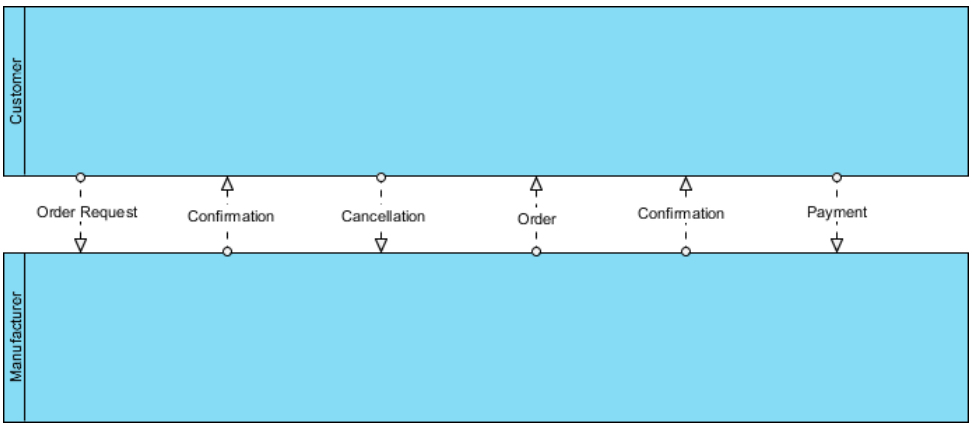
Jedním ze způsobů jak by šlo se podívat na model choreografie, tak představit ho jako znázornění smlouvy mezi dvěma nebo více business organizacemi. Následující obrázek 3.12 znázorňuje jednoduchý příklad procesu znázorněného pomocí choreografie. Uprostřed choreografické aktivity je uveden název aktivity, nahoře a dolu jsou účastníci interakci (účastníků může být víc než dva), iniciátor interakce je obarven stejnou barvou jako choreografická aktivita.



Obrázek 3.12: Příklad choreografie [4]

Kolaborace

Model kolaborace je sadou účastníků, které jsou znázorněny jako bazény (pools) a které komunikují mezi sebou pomocí toku zpráv (message flow). Kolaborace také může zahrnovat procesy uvnitř těchto bazénů a/nebo choreografií mezi těmito bazénami. Na následujícím obrázku je znázorněn příklad, jak by mohl vypadat model kolaborace. Je důležité uvést, že bazén může, ale nemusí být prázdný a může obsahovat procesy uvnitř.



Obrázek 3.13: Příklad kolaborace [4]

3.5 Porovnání BPMN s notaci UML

Protože BPMN není jedinou existující notaci na světě, v této podkapitole provedu její porovnání s další široce využívanou notací a ukážu rozdíly jejich účelů a využití.

UML (Unified Modeling Language) je jednou z nejznámějších notací, jejímž účelem je poskytnutí modelovacího jazyka pro znázornění designu softwarového systému. Obecně řečeno je to grafický modelovací jazyk pro specifikace, vizualizace a modelování artefaktů softwarového systému a, jako dopad, často se využívá při modelování softwarových systémů.

Na rozdíl od BPMN, UML je objektově-orientovaný jazyk, který nemá žádný popis toho, jak specifikovat či navrhovat SW systémy. Odsud lze říct, že UML není předpis, podle kterého by šlo navrhovat systémy, ale notace pro dokumentace designu. [24]

Jak je dobře vidět z tabulky, obě dvě notace mají svoje využití, však v různých sférách.

UML	BPMN
Grafický jazyk pro vizualizaci, specifikaci, návrh a dokumentace programových systémů	Grafická notace sloužící k modelování podnikových procesů, která by měla být jednoduše srozumitelná všemi business uživatelí
Většinou se používá pro modelování softwarových systému	Používá se pro modelování business procesů
Využivá objektově-orientovaný přístup pro modelovani aplikací	Využiva procesně-orientovaný přístup pro modelování systémů

Tabulka 3.1: Porovnání notací UML a BPMN

Modelováním s využitím notace BPMN (a dalších) se zabývají procesní analytici. Další podkapitola je věnována popisu toho, co procesní analytik je a jaké jsou očekávané kompetence na této pozice.

3.6 Procesní analytik

Obecně řečeno, procesní analytik je osoba odpovědná za analýzu podnikových (a často i IT) procesů a jejich modelování. V případě, že podnik ještě nemá namapované procesy, obvykle procesní analytik začíná svoje práce přesně procesním mapováním. [25]

Před tím až se pustím dál je potřeba vydefinovat procesní mapování pro lepší pochopení k čemu ono slouží. Podle Fialy [26], procesní mapování je definováno jako “disciplína procesní analýzy, která poskytuje nástroj a ověřenou metodologii k identifikaci stávajících procesů ve firmě (procesů „jak to je“) a lze využít jako návod pro zlepšování podnikových procesů (procesů „jak to má být“)”. Z toho lze říct, že jedná se o nástroj procesního řízení využívaný k lepšímu pochopení stávajících firemních procesů a jejich následné optimalizaci.

Mapováním procesu je činnost, která v sobě zahrnuje vytvoření mapy procesu. Není vždy nutné mapovat proces celý, ale určitě by měla být namapována část procesu, která nás nejvíc zajímá. Mapováním procesu získáme představu o tom co v procesu se děje a co do něho zasahuje. Obecně mapování procesů je založeno na metodě strukturní analýzy. Tato metoda požaduje splnění následujících podmínek pří mapování:

- Procesní mapa musí být vnitřně konzistentní
- Z procesní mapy musí být vidět, jaké činnosti proces vykonává

- Procesní mapa musí přehledně graficky znázornit prvky (objekty) a činnosti (vykonávané člověkem nebo strojem)

Nyní víme, co si přesně představujeme pod pojmem procesního mapování. Bylo by však vhodné zmínit se i o dalších kompetencích procesního analytika. Mezi nimi Ing. Polan [25] a většina inzerátů na tuto pozici na Internetu uvádí:

- **Komunikační dovednosti**. Schopnost komunikovat s lidmi je určitě důležitou podmínkou pro analytika. Při mapování a modelování procesů je nutné komunikovat se zaměstnanci odpovídajících úrovní, aby namodelovaný proces odpovídal realitě. Navíc analytik by měl umět dobře vykládat informace o výsledcích své práce pro vše členy business procesů, které s velkou pravděpodobností budou navazovat na této výsledky. Také by se očekávalo, že procesní analytik dobře rozuměl tomu, že různé skupiny business procesu od něj potřebují různé typy informací. Business by například určitě nezajímaly detaily implementace.
- **Odborné znalosti**. Samozřejmá podmínka, jinak bez těchto znalostí by procesní analytik ani nemohl dobře namapovat procesy, co by určitě mohlo přivést k neefektivnímu řízení firmy až její krachu v nejhorším případě.
- **Technické znalosti**. Protože procesní analýza a řízení jsou docela spojeny s IT, tak od zaměstnance na této pozice také očekávají schopnosti ovládat SW nástroje jako MS Office, Enterprise Architekt a vyplývající z toho znalostí modelovacích notací (např. UML).

Protože jednou z zodpovědností procesního analytika je tvorba modelu, v další podkapitole rozeberu příklad použití notace BPMN na poměrně jednoduchém procesu podání žádosti o zaměstnání.

3.7 Příklad modelování procesu v notaci BPMN

Pro lepší představu, jak by mohl vypadat proces, namodelovaný v notaci BPMN, uvádím jako příklad proces, se kterým v životě se setká skoro každý– podání žádosti o zaměstnání. Popis procesu v rámci firmy je následující:

Potenciální zájemce zašle svůj životopis včetně dalších potřebných dokumentů v rámci přijímacího řízení. Zodpovědná osoba či osoby—pravděpodobně HR oddělení—si prohlédne tuto žádost a v případě, že zájemce nevyhovuje, můžou žádost odmítnout, o čem ten dostane zprávu a tím proces je ukončen.

V opačném případě zájemce bude pozván na příjímací pohovory. Po provedení této činnosti dál HR oddělení (oddělení lidských zdrojů česky) bude rozhodovat o nabídnutí zájemci práce. V případě, že s pohovorem oni byli spokojený, dostane zájemce si nabídku práce, v opačném případě je jeho žádost zamítnuta a proces je ukončen stejně jako v případě odmítnutí žádosti v prvním kroce. Po obdržení nabídky práce si zájemce může rozhodnout, že není spokojený s podmínkami této nabídky a odmítnout jí, o čemž dostane zaměstnavatel zprávu a tím proces bude ukončen. V případě, že je zájemce s nabídkou spokojený, tak se podepíše smlouva mezi oběma stranami a firma má nového zaměstnance.

Na obrázku v příloze A.1 je znázorněno, jak takový proces by mohl být namodelován procesním analytikem v notaci BPMN. Tento model v sobě zahrnuje docela velkou míru detailu a v závislosti od požadavků vedoucího procesního manažera mohl by být o něco zmenšen.

Například, diagram by šlo zjednodušit za využití tzv. zatahnutého bázenu (collapsed pool). Z hlediska zaměstnavatele, nás spíš nebudout zájímat aktivity kandidáta/zájemce. Lze tento diagram zjednodušit využitím tzv. collapsed pool. Na dalším obrázku v přílozeA.2 je znazorněno jak by takový diagrám vypádal

3.8 Závěr

V této kapitole jsem popsal modelovací notaci BPMN. Začal jsem definicí a cíli této notace. Pokračoval jsem úvodem do historie vzniku notace a dále jsem podrobně rozepsal základní prvky, které se objevují v této notaci. V následné části jsem uvedl různé typy modelů, které se v této notaci vytváří a v informativních účelech jsem provedl porovnání BPMN s široce známou notaci UML. Na konci jsem krátce popsal profesi procesního analytika a uvedl jsem příklad namodelovaného pomocí BPMN 2.0 procesu z reálného života.

Kapitola 4

Nejčastější chyby BPMN modelů

V této kapitole provedu popis a rozbor chyb, vyskytujících se při modelování procesů v notací BPMN. Na začátku ukážu výsledky rešerše z externích zdrojů, včetně podrobného popisu jednotlivých chyb a analýzy jak těmto chybám by šlo předejít nebo jak by tyto chyby šlo opravit. Na závěr provádím analýzu všech identifikovaných chyb a popisuji, které z nich v rámci této práce nemá smysl řešit.

4.1 Úvod do problematiky analýzy modelů

Výhodou (a současně i nevýhodou) notace BPMN je to, že navrhnutý model vždy není jediné možné řešení, a notace umožňuje skoro nekonečné množství různých řešení stejného úkolu. Diagramy se mohou lišit, například mírou detailů, počtem účastníků nebo cílovou skupinou pro čtení tohoto diagramu. Je to přesně důvod, proč bych chtěl v tomto úvodu říct, že neexistuje stoprocentní způsob ověřit správnost modelu, pokud je ten model vytvořen podle přijatých standardů a metodik.

Procesní analytik musí být schopen, na základě analýzy, rozumět potenciálním chybám a umět jim vyhnout. Proto jsou v notaci metodiky, kterých by uživatelovi bylo doporučeno dodržovat. Obecně se jim říká "good practices" nebo správné postupy česky. Je velice doporučeno, aby se analytik snažil co nejvíc těchto rad dodržovat během modelování procesů. Protože pohledů na model může být hodně, tato práce se zabývá přesnou kontrolou dodržování těchto pravidel. Navíc, podle [5] neexistuje moc studií, které by se zabývaly problematikou analýzy často vyskytujících chyb v modelovací praxi. S tímto tvrzením souhlasím i já z důvodu, že při provedení rešerše tuto téma, většina studií které jsem si našel, odkazovaly přesně na citovanou práci [5]. Jediná studie, která se v minulosti zabývála tímto tématem, byla publikována v

roce 2008 [27], ale v rámci tohoto textu se jen provedla analýza nejčastěji vyskytujících trendu při modelování procesů.

4.2 Chyby procesních analytiků

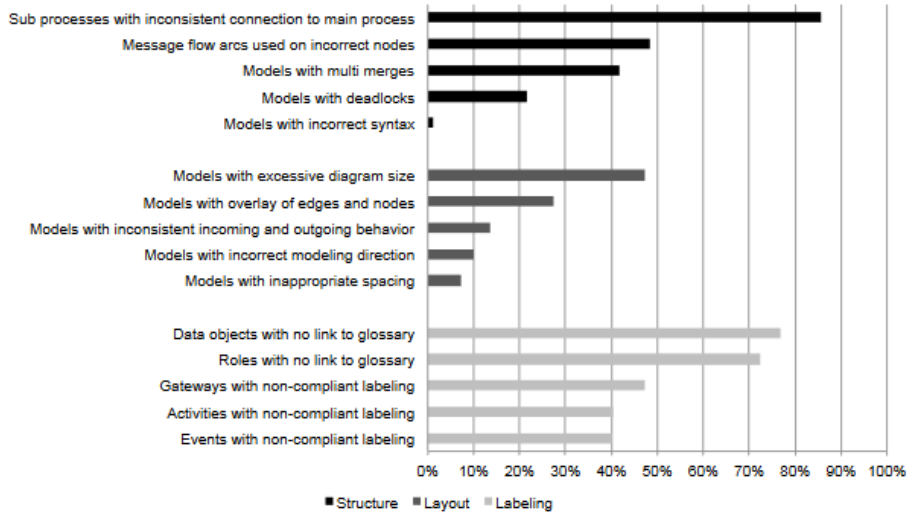
Chyby, popsané v této podkapitole lze rozdělit na tři podtypy:

- **Strukturální**. Tyto chyby jsou většinou spojeny s nekorektním využitím prvků notací nebo jejích propojením. Je potřeba klást velký důraz na chyby tohoto typu, protože jsou nejdůležitějšími chybami při tvorbě procesních modelů, a strukturální chyby komplikují průběh procesní analýzy.

- **Dispoziční**. Chyby tohoto typu jsou spojené s umístěním prvků v modelu. Nedbání toho, jak model vypadá, může mít takové důsledky jako špatná čitelnost modelu a jako dopad komplikuje to pochopení modelů ostatními analytiky.

- **Pojmenování**. Chyby v pojmenování prvků modelu nebo jejich popisu. Většinou tyto chyby jsou zanedbatlené, ale můžou působit konfliktům z hlediska konzistence modelu. Špatně pojmenované elementy v modelu mohou mít velký vliv na to, jak čitelný a transparentní model je. Dokonce to může mít vliv i na procesní analýz. Například, analytik může pochopit běh procesu jinak, než se myslelo autorem modelu. Je to jeden z důvodu proč je potřeba dodržovat základních pravidel, tímto způsobem lze takovýmto problémům předběhnout. Standardy obecně doporučují využití formátu "Sloveso-objekt" pro události (Zamítnout žádost), "Objekt-příčestí" pro aktivity (Žádost zamítnuta) a "Objekt-příčestí" pro brány ("Žádost zamítnuta?")–viz o prvcích notací BPMN v kapitole **"Modelovací notace BPMN"**.

Na následujícím obrázku 4.1 je grafické znázornění výsledků analýzy procesních modelů z praxi. Na tomto obrázku je seznam chyb, kterých procesní analytici v praxi dopouštěli co nejvíc.



Obrázek 4.1: Nejčastější vyskytující porušení modelovácích pravidel v praxi. [5]

Chyby, ke kterým se docházelo během modelování jsou rozděleny do třech již zmíněných skupin. Ve skupině strukturálních chyb:

- **S1**. Nekonzistentní propojení hlavních procesů a podprocesů. Tato nejčastěji vyskytující se chyba je spojena s tím, že analytici neberou v úvahu to, že model by měl být považován za část procesní architektury skrz celou firmu. Jinými slovy, při tvorbě modelu, který v sobě obsahuje subprocess, tento by měl být vykonán stejnou rolí jako hlavní proces.
- **S2**. Nekorektní využití toku zpráv. Tato chyba je spojená s propojením toku zpráv s elementy, se kterými by propojeny být neměly. Jednou z nejčastějších chyb je uvedena chyba, když analytik propojoval příchozí zprávu (incoming message) se zprávou házecí (throwing message) [1].
- **S3, S4**. Existence několika sloučení (multimerge) a deadlocků. Problém multi merge je většinou spojen se špatným využitím brán (gateway), kdy může dojít ke sloučení několika sekvenčních toku v elementu, který očekává průtok jedním tokem. Deadlock většinou je také spojen se špatným využitím brán a zastavuje proces nebo blokuje jeho pokračování v toku, což může vést k tomu, že proces nebude nikdy ukončen.
- **S5**. Nekorektní syntaxe. Tento typ chyb se vyskytuje v modelech, kde se došlo k nekorektnímu propojení elementů v modelu nebo v modelech, kde toto propojení chybí (i kdyby být mělo). Tohoto typu chyb se většinou dopouští začátečníci a proto nalezení tohoto typu chyb v 1 % zanalyzovaných modelů ukazuje na to, že se nejedná o něco, co by poměrně zkušený procesní analytik udělal.

[1] Házecí zpráva je autorský překlad throwing message do češtiny

Dispoziční chyby, které byly nalezeny během studií:

- **L1**. Příliš velký rozměr modelu. Model by se měl umístit na papíru formátu A3. V případě příliš velkého modelu je potřeba si rozmyslet pokud by nebylo vhodnější rozbít model na menší části (dekomponovat) nebo změnit rozměr modelu využitím podprocessů.
- **L2**. Překryvání se prvků v modelu. Podle standardu, je potřeba nedopouštět překrývání hranic prvků modelu. Nedbání tomuto vede ke zhoršení čitelnosti procesního modelu.
- **L3**. Nekonzistentní výchozí a příchozí toky. Toky zpráv a sekvenční toky by měly být konzistentně využívané ve stejných místech v diagramu. Znamená to například, že sekvenční tok by neměl začínat v dolní části aktivity.
- **L4**. Nekorektní směr modelu. Podle specifikace by model měl "téct" zleva doprava. To znamená, že sekvenční tok během procesu by v ideálu měl jít směrem doprava.
- **L5**. Nekorektní odstupy mezi elementy. Odstup mezi elementy by měl činit minimálně 50 % rozměru samotného elementu.

Chyby spojené s pojmenováním objektů lze rozdělit do dvou skupin:

- **G1, G2**. Spojené s nevyužitím glosáře [2]. Využití glosáře v modelu pomáhá předběhnout neporozuměním a nekonzistencím mezi procesní analytiky a zvětšuje čitelnost samotného procesu. Ze studií je vidět, že většina analytiků glosář nevyužívá (Datové objekty a role bez linků do glosáře). Pravděpodobně pro malé firmy tento typ chyb moc neovlivňuje, ale pro velké firmy s několika týmy procesních analytiků, upřesnění využívaného jazyka má velký přínos.
- **G3-G5**. Chyby spojené s nedodržením standardů pojmenování. Jak již jsem zmiňoval v popisu chyb, jedná se o využití správného slovního formátu pro pojmenování prvků v modelu.

Dál uvedu několik vlastních doporučení na základě seznámení s dokumentací notace BPMN. K těmto doporučením, na které by měl procesní analytik pamatovat, patří využití počátečních a koncových událostí (Start Event a End Event respektivě). Podle specifikace BPMN tyto prvky nejsou povinné v modelu, ale bez něj může v modelu dojít k neporozumění a poměrně silné

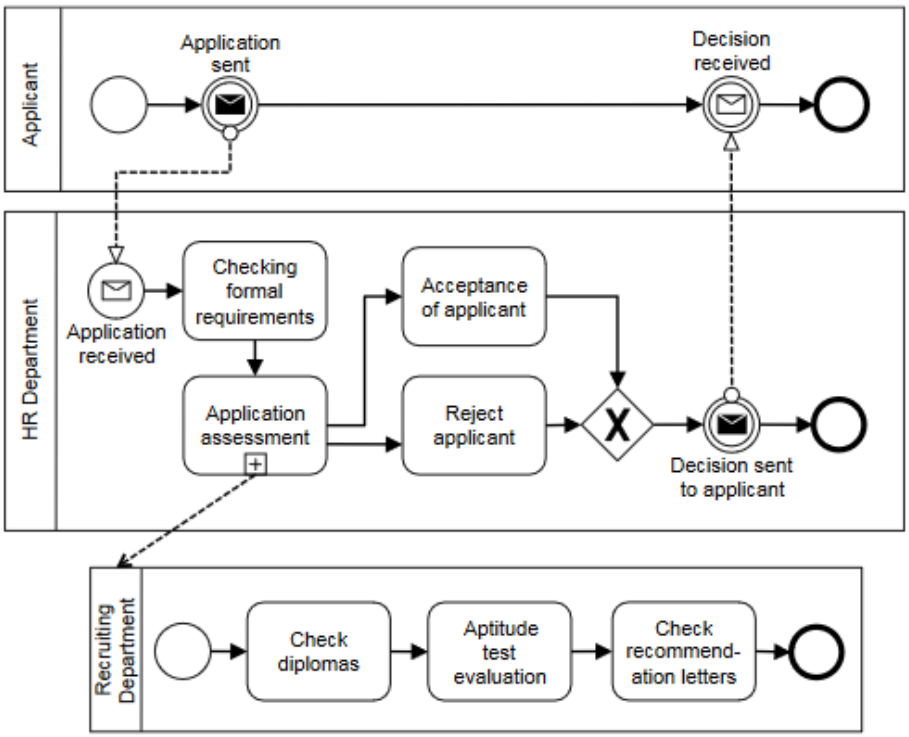
[2] Část modelu, kde je popsána terminologie využita v modelu. Využívá se většinou v podnikách s velkým počtem modelovaných procesů a zavádí praktiku využití stejného "jazyku" a pomáhá předběhnout neporozuměním mezi různé týmy analytiků.

se snižuje čitelnost modelu. K dalším doporučením, které se týkají těchto prvků patří důležité upozornění: v případě existence několika počátečních a koncových událostí v modelu je potřeba unikátně pojmenovat každou z nich pro jednoznačnou interpretaci modelu.

Jako další doporučení je potřeba správného využití prvků podle kontextu. Někteří analytici chybně používají konečnou událost(End event) a přerušující konečnou událost (Terminate end event). Přerušující konečná událost ruší vše běžící toky v modelu narozdíl od koncové události, která ukončuje jen jeden tok v procesu. Většinou se doporučuje využít přerušující koncovou událost pro označení výjimečného dopadu během procesu.

4.3 Rozbor chyb a způsoby jejich oprav

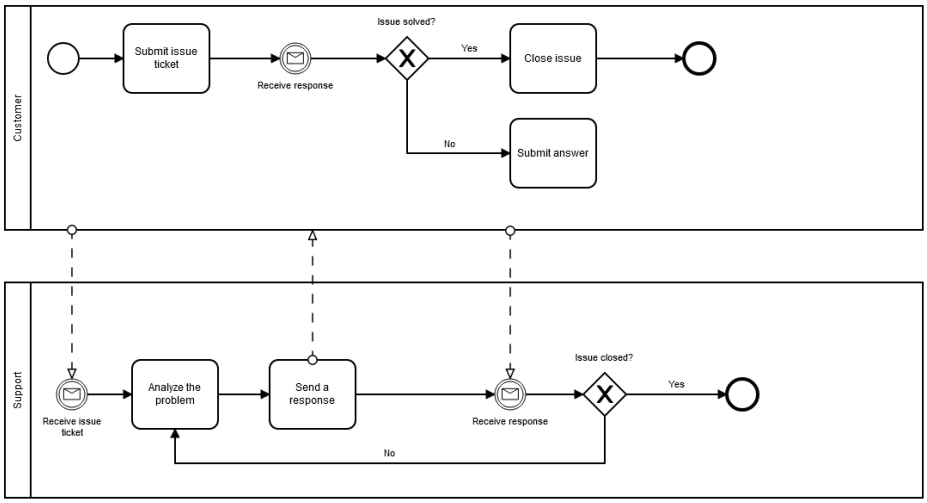
Na obrázku 4.2 je ukázán příklad výskytu chyby **S1** v modelu. Jak již bylo zmíněno, problém je v nekonzistenci rolí, které procesy řídí-hlavní proces je řízen roli HR Department, přičemž subprocess je řízen roli Recruiting Department.



Obrázek 4.2: Nekonzistentní propojení hlavního procesu a podprocesu. [5]

Jako řešení by byla potřeba buď zmenšit míru detailu a nezobrazovat tento podproces nebo označit roli zodpovědnou za podproces stejnou jako roli, která koná hlavní proces - Application assessment. Znamená to, že tento podproces je potřeba mít uvnitř bazénu "HR Department".

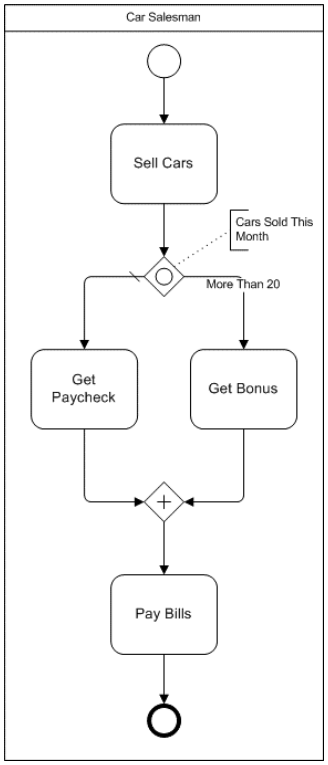
Na obrázku 4.3 je zobrazen příklad chyby stejné s typem chyb **S2**. V modelu je vidět, že toky zpráv jdou přímo z bazénu Customer. Dopouštění chyby, když přichází zpráva (incoming message z aktivity nebo throwing message event) je odeslána do činností zprávu házející (throwing message event), ve většině modelovacích nástrojů není povoleno na implementační úrovni, proto příklad této chyby tady neuvádím.



Obrázek 4.3: Nekorektní tok zpráv

Chyba na obrázku 4.3 se opravuje tím, že tok zpráv správně je odeslán buď z události, nebo z činností zprávu házející (v modelu nejsou), nikoliv z bazénu. Hlavní problém takového modelu je, že z něho nedá se moc poznat o tom, jak přesně se probíhá komunikace mezi stranami (bazény).

Na obrázku 4.4 je ukázka BPMN modelu, který obsahuje deadlock, který odpovídá typu chyb **S4**.

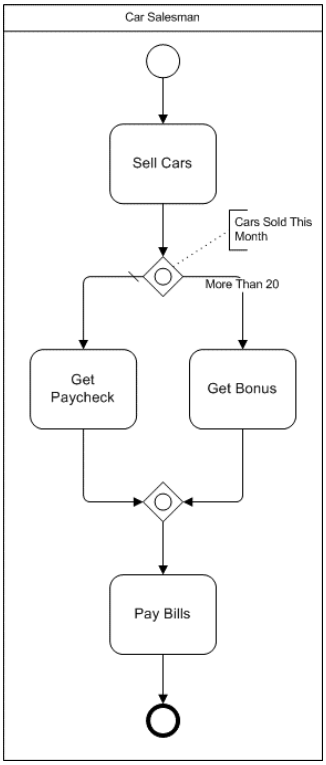


Obrázek 4.4: BPMN Deadlock příklad.[6]

Na první pohled se zdá, že tento diagram je v pořádku, ale není to tak. Zásadní problém leží ve využité spojovací bráně po rozdělení procesu. Toto je paralelní brána (parallel gateway), podle specifikace tento typ brány potřebuje, aby vše rozvětvení byly provedeny, aby mohl proces pokračovat. V tomto případě ale existuje situace, kdy brána bude nekonečně čekat na ukončení všech aktivit v procesu.

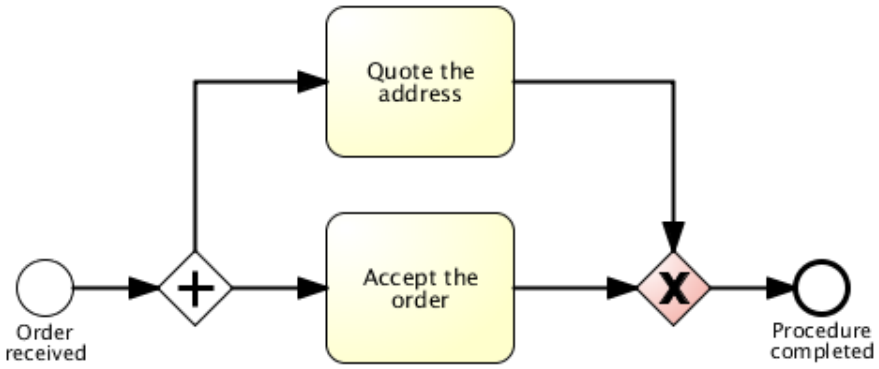
Proces se dělí na dvě cesty pomoci inkluzivní brány OR (tok může projít všemi cestami, na které se dělí, ale nemusí). V případě, že prodejce prodal více než 20 aut, tak v tomto diagramu nedojde k deadlocku–seknvenční tok projde oběma cestami, čímž splní podmínku, na kterou paralelní brána čeká (vše sekvenční toky musí uspešně "dojít"do brány). Avšak, až nastane situace, kdy bylo prodáno méně aut, než 20, tak jedna z cest už nebude platnou. Paralelní brána toto bohužel, neví a bude stále čekat na ukončení aktivity na cestě, na kterou procesní tok už se nedostane.

Řešením této situace je změnit zvolenou paralelní bránu za inkluzivní bránu OR, která bude čekat jen na větve, které jsou aktivní. Tento diagram je znázorněn na obrázku 4.5.



Obrázek 4.5: BPMN Deadlock oprava diagramu.[6]

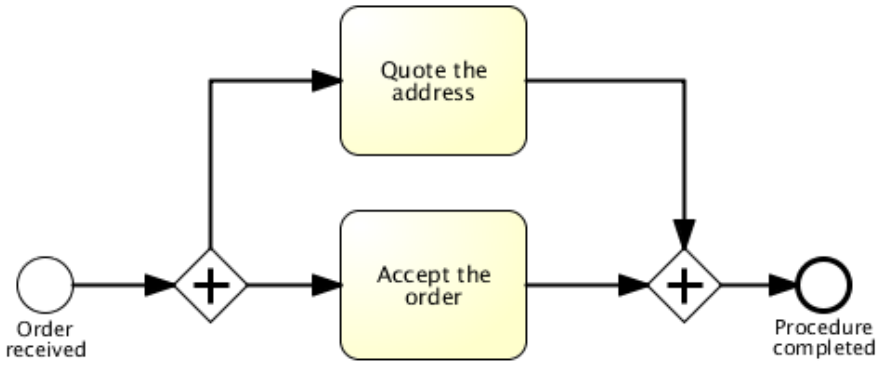
Na obrázku 4.6 je příklad multi merge. Stejně jako s deadlockem, jedná se o nekorektní využití brán. I když se zdá, že je diagram v pořádku, brána napravo není správného typu. Je to exkluzivní brána AND, která očekává jeden vstup, ale v tomto příkladu dostává dva, což vede k nedefinovanému chování.



Obrázek 4.6: BPMN Multimerge Problem.[7]

Jednoduchý způsob opravy tohoto modelu je využití inkluzivní brány OR,

pomocí které zbavíme se problému, že do prvku, který očekává jeden vstup, dostáváme vstupů víc. 4.7



Obrázek 4.7: BPMN Multimerge Problem Solution.[7]

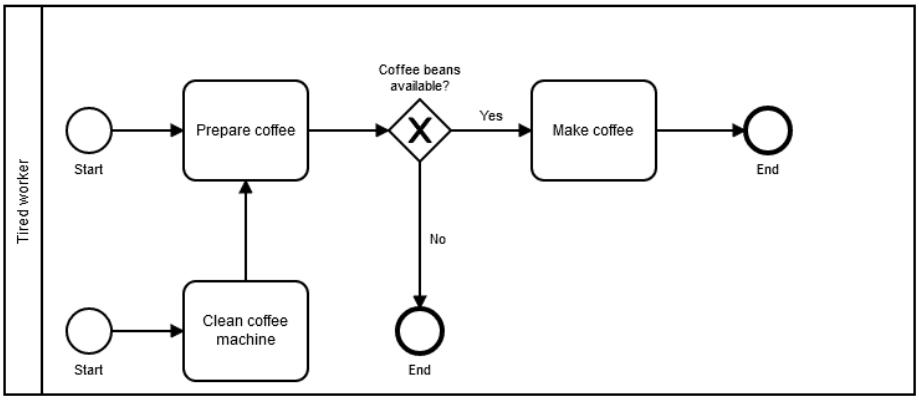
Na dalším obrázku 4.8 je ukázky chyb typu **L1**, **L3** a **L4**. Protože to jsou chyby dispozičního charakteru, lze jím předběhnout dodržováním specifikace, detaily ohledně kterých jsou popsány v podkapitole **"Chyby procesních analytiků"**.

Obrázek 4.8: BPMN Inconsistent Layout Example.

V tomto modelu typu chyby **L1** odpovídá ten fakt, že celý model je zbytečně protáhlý. Určitě by vešel na papír formátu A3, ale ten diagram by šlo o něco zkrátit (z hlediska délky sekvenčního toku z aktivity "Meet up with a neighbor" do aktivity "Go home"). Typu chyb **L3** odpovídá ten fakt, že sekvenční toky ochazející z jednotlivých aktivit odchazí vždy z jiného místa: sekvenční tok z aktivity "Go for a walk" odchází z dolní hranice obdélníku, sekvenční tok z aktivity "Meet up with a neighbor" odchází z horní hranice

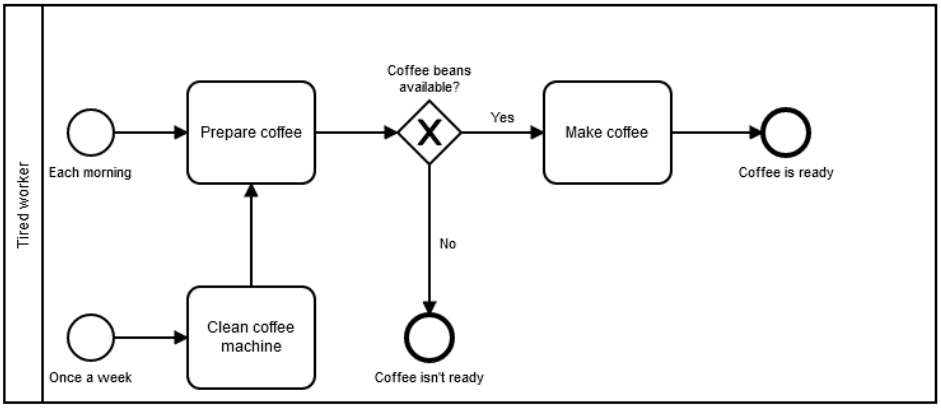
obdélníku a sekvenční tok z aktivity "Come back home" odchází z pravé hranice obdélníku. Typu chyb **L4** odpovídá ten fakt, že model zbytečné "teče" na začátku v dolním směru, pak pokračuje směrem nahoru a nalevo. Jak je vidět, to jsou problémy přehlednosti modelu, ale je analytikům doporučeno tímto problémům vyhnout.

Na obrázku 4.9 je znázorněn příklad využití několika počátečních a koncových událostí. Problém není vidět úplně ihned, ale obě dvě počáteční a koncové událost jsou pojmenované stejně, co může působit nepochopení modelu.



Obrázek 4.9: BPMN Multiple Start Events.

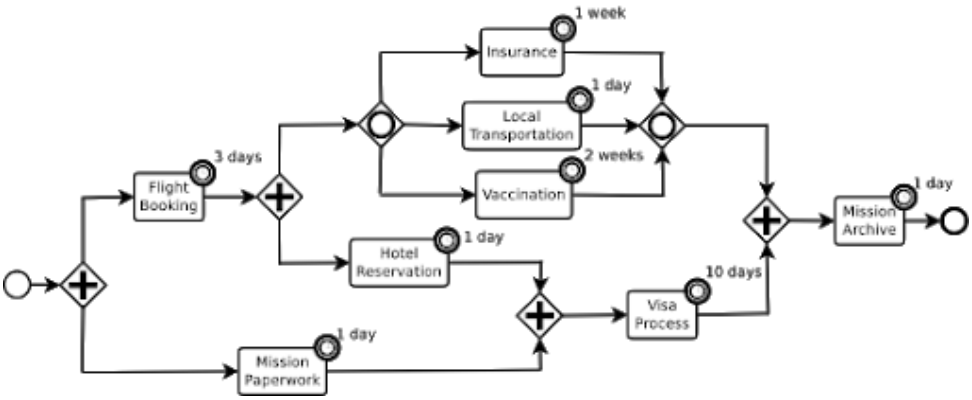
Pro lepší znázornění toho, co se děje v procesu je potřeba přejmenovat této události, jako je ukázáno na obrázku 4.10. Jak již bylo zmíněno, tato oprava usnadňuje pochopení procesu a likviduje možné nepochopení běhu procesu jinými analytiky.



Obrázek 4.10: BPMN Multiple Start Events Corrected.

Od sebe to této kapitoly přidal bych problém paralelismu, který je spojen

s deadlockem a multi mergem (chyba typu **S3, S4**) a jako ukázka je na obrázku 4.11 příklad procesu, během kterého aktivity se dělí na několik paralelně probíhajících a každá musí být ukončena. V případě, kdyby analytik zapomněl na ukončení jednoho z probíhajících paralelně procesů, tj. buď přidání konečné události pro cestu nebo propojení s bránou správného typu, došlo by k chybě typu **S3** v modelu.



Obrázek 4.11: BPMN Parallelism. [8]

4.4 Analýza relevantnosti odhalených chyb

Protože chyb, ke kterým může dojít během modelování je skoro nekonečné množství, je potřeba z pohledu této práce mírně "odfiltrovat" je podle jejich relevantností. Tato analýza je důležitou součástí práce, jelikož na jejím základě bude popsán návrh a požadavky na funkčnost aplikace (viz kapitola **"Požadavky na aplikaci"**).

Chyba typu **S1** nemá moc smysl řešit v rámci této práce, protože se zabývá tzv. high-level modelováním, kde každý model je uvažován jako část velkého systému nebo firmy. Tato práce uvádí tuto chybu jako něco, co by procesní analytik měl mít na mysli, ale na základě diskuze s vedoucím a vlastního rozhodnutí nebude v rámci této práce řešena. Jako další důvod bych uvedl to, že většina studentů ani nedopouští této chyby z toho důvodu, že v škole nemodelují se procesy velkých obchodníků nebo podnikatelů.

Za důležitý považuji popis chyby typu **S5**. Protože tato práce je navázaná na práci, která se zabývala vzděláváním a výukou procesních analytiků, předpokládá se úroveň základních znalostí z procesního modelování. Většinou aplikace očekává model, který dává smysl a by mohl být namodelovan někým, kdo již nějakou zkušenost s modelováním v notací BPMN měl. Přesto v případě nalezení syntaktických chyb, které aplikace kontroluje, bude ona snažit uživateli napovědět. V případě základních neznalostí může nastat situace, kdy aplikace uživateli nepomůže–buď se něco v aplikace rozbije a

kontrola neproběhne nebo aplikace nebude schopná chybu odhalit. V tom případě je uživateli doporučeno využit odkáz na dokumentaci od OMG.

Chyby dispozičního charakteru, které považuji za důležité v rámci kontroly jsou **L1**, **L2** a **L4**. Kontrola toku zleva napravo zavádí dobrou praktiku pro procesního analytiku dodržovat standardu, a nepřekrytí prvků v modelů je důležitou součástí každého namodelovaného procesu. Problém typu L1 bude taky uvažován, ale nebude se počítat za zásadní chybu.

Chyby spojené s pojmenováním v rámci této práce nebudou řešeny, jelikož jsou méně důležité, než problémy spojeny s dodržováním správné syntaxe modelů. Chtěl by však uvést, že rozšíření aplikace o slovník, podle kterého by se kontrolovalo dodržení přijatých konvencí spojených s pojmenováním, by mělo pokrýt většinu problémů, se kterými se setkávají studenti při pojmenování elementů v modelu.

Jako závěr této podkapitoly bych shrnul, že v rámci této práci za nejrelevantnější chyby se většinou považují skupinu strukturálních chyb (je popsána v podkapitole "Chyby procesních analytiků"). Součástí této skupiny jsou i chyby syntaktického charakteru jsou to chyby, ke kterým docela často se dochází a na které studenti během studia předmětů jako “Základy procesního řízení” jsou nejvíc upozorněny.

4.5 Závěr

V této kapitole jsem se zaměřil na objasnění a popis chyb, ke kterým dochází v praxi během procesního modelování. Na začátku jsem se věnoval popisu problematiky procesní analýzy a uvedl jsem důvody proč kontrola správností modelů je docela subjektivní aktivitou. Následně jsem vypsal vše nalezené chyby, kterých se dopouštěli analytici v praxi a popsal jsem jejich význam a možnosti jak jim předejít nebo je opravit.

Neopomněl jsem přidat praktické příklady pro lepší názornost jednotlivých chyb. K seznamu chyb jsem taky dodal řadu doporučení, kterými by se dobrý procesní analytik měl řídit. V poslední podkapitole jsem shrnul všechny probrané chyby a diskutoval nad jejich relevantností vůči této práci a jejím účelům. V další kapitole se již budu věnovat popisu požadavků na aplikaci, která by umožnila studentům ověřit správnost nakreslených jimi modelů podle definovaných předpisů.

Kapitola 5

Požadavky na aplikaci

Obsahem této kapitoly je definice požadavků na aplikaci, včetně definování jednotlivých typů chyb, detekcí kterých aplikace by umožnila zájemcům o procesní řízení prakticky otestovat svoje znalosti z modelování procesů v notaci BPMN.

5.1 Účel aplikace

Hlavním účelem aplikace je poskytnout nástroj, pomocí kterého by zájemci o modelování v notaci BPMN mohli provádět rychlou analýzu jimi vytvořených modelů a vyhnout se výskytu nejčastějších chyb ve svých modelech. Tento software by mohl být využíván studenty pro srovnání s jimi získanými teoretickými a praktickými znalostmi. V případě dobrých dopadů z testování a průchod potřebnými kontrolami, využití této aplikace by mohlo byt zahrnuto do výuky předmětů jako "Procesní řízení".

5.2 Požadavky na obsah aplikace

Uživatel pomocí nástrojů aplikace nakreslí libovolný procesní model v aplikaci a pomocí interaktivního tlačítka ověří správnost tohoto modelu. Na základě provedené analýzy v kapitole **“Nejčastější chyby BPMN modelů”** vydefinoval jsem seznam chyb, které aplikace by měla schopna detekovat. Na moment napsání práce tento seznam vypadá takto:

- **Detekce deadlocků a multi merge situací, které vznikají kvůli nekorektnímu využití brán**. K tomuto typu ověřování také patří

nekonečné smyčky způsobeny využitím nekorektních typů brán a multi merge situaci spojené s nekorektním využitím rozcestí (forking) z aktivity.

- **Detekce chybějících odchozích nebo příchozích toků v modelu**. Například, aplikace je schopna detekovat element v modelu, který nemá odchází nebo příchozí tok, což je zásadní chybou v modelování.
- **Detekce nekorektní úrovně detailů při modelování kolaborace**. Každý účastník kolaborace musí být modelovan na stejné úrovní detailu. Tím se myslí, že v kolaboraci každá plavecká dráha obsahuje uvnitř sobě buď nic nebo namodelovaný proces.
- **Detekce neukončených toků v modelu**. Tímto se rozumí schopnost odhalit chybu, kdy model obsahuje několik počátečních událostí, ale ne všechny mají událost konečnou.
- **Detekce stejných jmen počatečních a koncových událostí**. Tady se jedná o konceptuální chybu, protože proces nemůže mít dvě stejně pojmenovaných počátečních nebo koncových činnosti.

Aplikace zkontroluje zda model neobsahuje žádnou z výše uvedených chyb a pokud model žádnou neobsahuje, bude uživatel informován o správnosti modelu. V opačném případě, se v modelu obarví problematické části a uživatel dostane krátký slovný popis všech nalezených chyb. Pokud uživatel bude chtít, tak bude moct opravit model a provést další kontrolu. Většina popisů chyb obsahuje odkaz na dokumentace od společnosti OMG zodpovědné za vývoj BPMN a uživatel bude mít možnost si nastudovat potřebnou informaci. Celkově by uživatel neměl strávit víc než 30-45 minut nad jedním modelem. Tento časový odhad uvažuje to, že libovolný model v sobě může mít několik chyb. V případě úplně korektního modelu uživatel by mohl mít hotovo i za méně než 15 minut–čas se liší v závislosti od rozměru a složitosti modelu.

5.3 Cílová skupina

Jako cílovou skupinu této aplikace vidím buď zájemce o modelování procesu v notaci BPMN nebo studenty, kteří studují předmět spojený s tímto tématem. Očekává se přítomnost základních znalostí spojených s procesním modelováním. Například uživatel, který prošel aplikací “Virtuální příprava procesních analytiků” [25], by mohl využít tuto aplikaci pro pokračování ve vylepšení svých dovedností spojených s modelováním v BPMN 2.0. Protože aplikace informuje uživatele o tom, kde a jaká chyba byla nalezena, má tato aplikace i výukový účel a se zvýšením stráveného v aplikaci času bych očekával, že uživatelé by začali vytvářet kvalitnější modely, které odpovídají mezinárodně přijatým standardům.

5.4 Technické požadavky

5.4.1 Obecné požadavky

Hlavním požadavkem na tuto aplikace je její univerzálnost a jednoduchost v jejím provozu a spuštění. Proto je potřeba využití moderních technologií, které budou stále aktuální i v dalších letech nebo budou stále podporovány s potřebou minimálních úprav. Kromě spuštění a provozu by měla aplikace mít intuitivní uživatelské rozhraní, se kterým by mohl bez problému interagovat co největší počet uživatelů.

5.4.2 Využité technologie

Na základě požadavků popsaných v předchozí podkapitole, je vhodným řešením naimplementovat webovou aplikaci, která bude přístupná z webového prohlížeče a nebude potřebovat další externí technologie jako databáze nebo další backend logiku. Stejně jako kolega Ing. Jan Polan [25], na práce kterého tato práce navazuje, jsem si rozhodl využít jeden z JavaScript frameworků.

HTML

HTML nebo HyperText Markup Language je značkovací jazyk široce využívaný pro vytvoření webových stránek. Vše prohlížeče tento jazyk snadno interpretují a je potřeba využit tento jazyk pro definice vzhledu aplikace. Poslední vydaná verze je HTML 5.

CSS

CSS nebo Cascading Style Sheets je jazyk, který se využívá pro vylepšení grafické reprezentaci prvků na HTML stránkách. Vše moderní webové stránky využívají HTML a CSS spolu, a CSS je požadavkem, pokud bychom chtěli, aby naše aplikace byla vizuálně atraktivní. Bez využití CSS by stránka měla původní vzhled HTML, což by bylo pro většinu uživatelů nepříjemně.

Bootstrap

Bootstrap je populární a známou knihovnou pro vytváření webových stránek. Je uživatelsky přívětivá a pro své základní funkci využívá knihovnu jQuery, která je taky široce využívanou knihovnou ve světě webových aplikací. Zvolil jsem bootstrap z důvodu její velké flexibility a předchozí zkušeností práce s touto knihovnou.

JavaScript

Jak již bylo zmíněno, je potřeba vyvíjet aplikaci pomocí moderních technologií, které ještě budou využívány v budoucnu. Moderní framework Vue.js splňuje danou podmínku. Kolega Zakhutskyi[28] provedl ve své práci srovnání tohoto frameworku s několika alternativy (a to jsou ReactJS, na kterém je postávena práce kolegy Polana [25], a AngularJS). Neměl jsem však velkou zkušenost s žádným z těchto frameworků a souhlasím, že docela podrobně napsaná dokumentace byla dobrou motivací pro výběr této knihovny. Podle výzkumu je druhým nejpoužívanějším frameworkem po populárním React.js, který je spravován společností Facebook. Velké využití tohoto frameworku po takovém velkém konkurentovi ukazuje na spolehlivost a kvalitu daného výběru.

V této kapitole jsem vydefinoval požadavky na aplikaci, která by umožňovala studentům a zájemcům pomoct dodržovat přijatých standardů a norem v modelovací notaci BPMN. V další kapitole podrobněji popíši její funkcionalitu a logiku.

Kapitola 6

Náplň a logika aplikace

V této kapitole se zabývám popisem náplně aplikace, a její logiky včetně popisu využitých algoritmů.

6.1 Náplň aplikace

Aplikace v sobě poskytuje kreslící nástroj umožňující modelovat procesy v notaci BPMN a sadů metod pro ověření správností modelu. Protože kreslící nástroj je implementován s využitím externí knihovny a není hlavním výsledkem této práce, je popis této součástí aplikaci uveden v kapitole **“Obsah a vývoj aplikace”**. Proto v této podkapitole popíšu druhou část aplikace, a to detekce chyb v modelu. Samotné typy chyb, které aplikace bude řešit, již jsem popsal v kapitole **“Požadavky na aplikace”**.

Však uvedu několik detailů ohledně toho jak aplikace s uživatelem pracuje. Aplikace neprovádí kontrolu během modelování, ale jen po obdržení žádosti od uživatele. Je to koncipováno takovým způsobem ze dvou důvodů: první je uživatelská přívětivost. Podle mého názoru zobrazování chyb ve grafu hned by bylo intruzivní a také možná náročnější na systémové prostředky z důvodu počtů prováděných kontrol. Druhým důvodem je to, že takový požadavek byl definován v zadání podle kterého tato aplikace se vyvíjela.

Přesně proto uživatel dostane od aplikace zpětnou vazbu jen po vyžádání provedení kontroly modelu. V další podkapitole krátce provedu popis chyb, které byli vydefinovany v kapitole **“Nejčastější chyby BPMN modelů”**, ale nedošlo k jejich implementací.

6.2 Technická omezení

Jako chybu, která byla v studii [5] označena jako jedná z časté vyskytujících a kterou jsem chtěl implementovat, bylo nekorektní využití toků zpráv. Například, kdyby odchází zprávny tok buď z činnosti nebo z chytací činnostní zprávy (Message Catch Event) šel do házecí činnostní zprávy (Message Throwing Event). Namodelovat takovou situací pomocí kreslícího nástroje od bpmn.io ani není možné. Skoro vše chyby takovéto typu, kde by analytik propojil špatným směrem elementy, které propojeny být ani nemohou, v této knihovně již zabraněny. Jako další základní příklad této chyb bych uvedl blokování možností provést sekvenční tok z aktivity do počáteční činnosti.

Nevyužití správného směru modelu zleva doprava bylo jednou z dalších chyb, které se vyskytovaly v praxi, ale relativně malý procent výskytu tohoto problému vůči dalším a ten fakt, že kreslící nástroj v knihovně bpmn.io již snáží kreslít diagram zleva doprava pokud využíváte automatické spojení, vedlo k vynechání tohoto problému z seznamu těch nejdůležitějších, které je potřeba implementovat.

V následující podkapitole popíšu podrobně, jak aplikace detekuje jednotlivé chyby a dává sebranou po kontrol informací dohromady.

6.3 Logika aplikace

Pro každou chybu popsanou v předchozí podkapitole, musela v aplikace vzniknout metoda, která by příslušný typ chyb detekovala. Před tím až se do popisu jednotlivých kontrol pustím, bych chtěl zmínit, že pro realizace všech z nich jsem využíval poskytnuté knihovnou bpmn.io (viz o této knihovně popíšu v kapitole **“Obsah a vývoj aplikace”**) atributy elementů v modelech této notace. Každý prvek v procesním diagramu má sadu atributu, ze kterých lze dostat informaci o objektu a objektech s ním spojených. Například, pro zjištění všech odchozích toků z elementu, je vhodné se kouknout do obsahu atributu “outgoing”, který je polím všech odchozích toků z tohoto elementu.

Kontrola deadlocků a multi merge

V rámci této metody probíhá kontrola správnosti využitých brán (Gateway) v modelu. Pro každou bránu, ze které odchází několik sekvenčních toků, v modelu metoda projde cestu až do buď další brány nebo až do koncové činnosti (End Event). Následně se provede kontrola typu brány, kterou cesta se začála, a konečné destinace, tj. buď další brány nebo koncového eventů. Pokud bude nalezena taková kombinace, která tvoří multi merge (například, několik cest z

paralelní brány se spojí v bráně exkluzivní) nebo deadlock (například, několik cest z inkluzivní brány se spojí v bráně paralelní), bude brána zdroje označena v modelu červeně a do seznamu chyb bude přidán záznam, obsahující popis chyby. V případě nenalezení žádné chyby, tato kontrola je považována za úspěšnou. Problém spojený s multi merge chováním také kontroluje i další metoda, kterou popíšu v příštím odstavce.

Kontrola nepropojených elementů

V rámci této metody se ověřuje pokud v modelu nejsou elementy, které nemají žádné příchozí a odchozí připojení. Pokud ano, jedná se o chybu v modelu, protože každý element za výjimkou počátečních a koncových událostí, musí mít jak odchozí, tak i příchozí tok. Počáteční a koncové události v rámci této metody se ověřují jednosměrně, tj. z každé počáteční událostí musí vést jeden sekvenční tok a nesmí existovat koncová událost, do které nevede žádný z toků.

Kontrola počtu odchozích a příchozích toků

V rámci této metody se ověřuje pokud počet odchozích toků jednotlivých prvků v modelu nepřevyšuje počet očekávaných. Obecně řečeno skoro vše elementy za výjimkou brán nemohou mít několik odchozích sekvenčních toků. Pokud nejedná se o bránu nebo aktivitu, vše elementy, které mají několik odchozích sekvenčních toků, tuto kontrolu neprojdou. V případě aktivity, kdyby aktivita měla několik odchozích sekvenčních toků, tak je potřeba navíc provést kontrolu, pokud nejedná se o rozcestí v grafu (forking). Tato kontrola se provádí pomocí porování souřadnic elementů, do kterých každý z odchozích toků teče. Je důležité říct, že toto porovnání souřadnic se provádí jen na ose x. Další důležitá poznámka je, že pro označení aktivity jako chybné stejně musí platit pravidlo několika odchozích sekvenčních toků. Jako příklad, v případě, kdy aktivita má odchozí jeden sekvenční tok a jeden tok zpráv chyba nalezena nebude. Má-li tato aktivita další sekvenční tok do nějaké jiné aktivity a destinace těchto toku nebudou na stejne ose x, aplikace to bude chápat jako chybu uživatele a upozorní ho na něj v modelu obarvením červeně sekvenčních toků odchozích z této aktivity.

Kontrola správné míry detailu v kolaboracích

Pokud v nakresleném procesu se jedná o kolaboraci, dana kontrola ověřuje pokud byla uživatelem dodržena korektní míra detailů. Tímto se myslí přesně to samé, co jsem již vydefinoval v předchozí podkapitole. Aplikace ověřuje pokud účastníci kolaborace mají potomky, tedy elementy uvnitř své plavecké dráhy (Swimlane). Pokud ano a existuje mezi účastníky tok zpráv, tak bude uživatel varován o nekorektní míře detailů a bude mu nabízené buď se zbavit namodelovaného procesu uvnitř účastníka (případně účastníků) s větší mírou detailů nebo namodelovat proces uvnitř dalších účastníků a nakreslit toky

zpráv mezi elementy uvnitř jednotlivých účastníků. V opačném případě problém v diagramu není a tento krok celkové kontroly je úspěšný.

Kontrola neukončených toků v modelu

Zodpovědností této metody je kontrola správností jedné ze základních vlastností každého modelů: každá počáteční činnost musí být někde ukončená. V rámcí této kontroly se porovnává počet počatečných (Start Event) a koncových (End Event) událostí a v případě, že počet koncových je menší počtů počátečních jedná se o neukončený tok v procesu. Pro lepší vyhledávání bude uživatelovi v modelu vše počáteční činnosti obarveny červeně pro snadnější vyhledávání problému.

Kontrola stejných jmen u počátečních a koncových událostí

Tato metoda ověřuje přítomnost několika počátečních a koncových událostí v modelu. Pokud v modelu jsou, provádí jednoduchou kontrolu, pokud nějaká případně nějaké počáteční událost případně událostí nemá stejné jméno jako nějaká další. Proces ověřování jmen koncových událostí je identický popsanému. Pokud identická jména budou nalezeny, budou v modelu této elementy obarveny červeně a uživatelovi v modálním okně se zobrazí hláška upozorňující ho na tuto chybu. V opačném případě

Po zmáčknutí tlačítka "Check"vše tyto kontroly se postupně provádí a případné popisy chyb se ukládají do pole proměnných errors. V případě, kdy žádná chyba nalezena nebyla, uživateli se zobrazí modální okno s gratulací, jinak uživateli se zobrazí modální okno, kde budou textově popsány všechny nalezeny chyby. Praktická ukázka průběhu práce s aplikací je popsána podrobněji v kapitole **“Obsah a vývoj aplikace”**.

6.4 Závěr

V této kapitole jsem popsal o jaké typy kontrol se v aplikací jedná, podrobně jsem popsal princip každé z nich včetně podmínek, které je potřeba splnit pro úspěšný průchod jednotlivými kroky. Neopomněl jsem krátce zmínit o typech chyb, které v rámci této aplikace jsem neřešil z důvodu jejich již existujícího řešení ve využité knihovně. Kapitola další se bude věnována popisu obsahu aplikace a detailům vývoje.

Kapitola 7

Obsah a vývoj aplikace

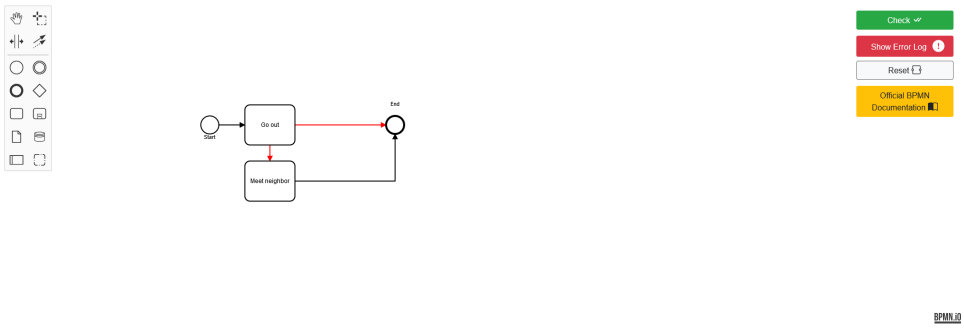
V této kapitole budu se věnovat popisu funkcionálního obsahu aplikace a provedu základní popis využitých technologií a některých technických detailů. Rovněž v poslední části uvedu možnosti rozšíření aplikace v budoucnu.

7.1 Průběh práce s aplikací

Aplikace vytvořena v rámci této práce dovoluje uživatelům nakreslit diagram v notaci BPMN a následně provést kontrolu jeho správnosti. Po spuštění uživatel dostane ihned do kreslícího prostředí, kde bude mít možnost namodelovat proces, který si přeje ověřit. Kreslící nástroj, který odpovídá za činnost kreslení, je poskytnut platformou Camunda a je realizován za využití knihovny bpmn io (viz podkapitola "Práce s knihovnou bpmn.io"). Na obrázku 7.1 je ukázka počáteční obrazovky, kterou je přivítán uživatel po spuštění aplikace.

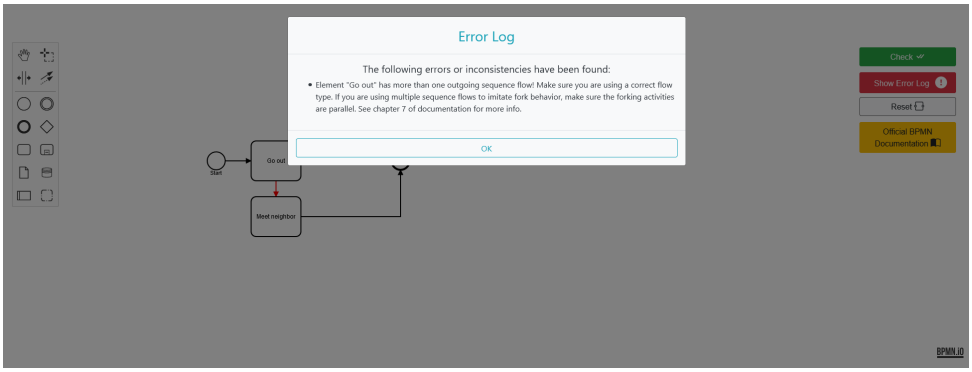
Protože aplikace předpokládá, že uživatel již má zadání či proces, který má namodelovat, není v aplikaci rozhraní, které poskytuje informace o tom, jaké zadání uživatel má-toto by měl již vědět uživatel sám. Požadavky k cílové skupině uživatelů byli popsány v kapitole **"Požadavky na aplikace"**. Předpokládá se, že uživatel je schopen samostatně vypracovat model. Po té až uživatel namodeluje proces, musí dát aplikaci vědět, že chce provést kontrolu modelu. To se dělá zmáčknutím zeleného tlačítka "Check". Po provedení této akci se uživateli zobrazí modální okno, obsahem kterého bude buď gratulace s uspěšným průchodem kontroly nebo slovní popis všech nalezených chyb, včetně odhadu jejích původu. V případě nalezení chyb, oni se také objeví v diagramu prostřednictvím obarvení chybných elementů nebo spojení jak je zobrazeno na dalším obrazku 7.2.

Obrázek 7.1: Zobrazená kreslící plocha po spuštění aplikace.



Obrázek 7.2: Jednoduchý příklad zobrazení chyb v diagramu.

V případě, kdyby uživatel zapomněl na vysvětlení nebo by chtěl z nějakého důvodu si přečíst slovní popis chyb ještě jednou, pomocí tlačítka "Show Error Log"bude moct zavolat stejné modální okno, které se mu zobrazovalo po provedení původní kontroly. Slovní popis chyb je postaven tak, aby snažil uživateli dát co nejvíc informace, včetně odhadu příčiny chyby a způsobu jejího odstranění. Následující obrázek 7.3 obsahuje příklad, jak vypadá toto modální okno po provedení kontroly modelu z předchozího obrázku.



Obrázek 7.3: Modální okno s popisem chyb po provedení kontroly.

Po případné opravě všech chyb, uživatel znovu má možnost provést kontrolu správnosti jeho nebo její modelu. Tento proces může se opakovat do toho, až uživatel bude spokojen s výsledky modelování nebo až nebude jeho nebo její model obsahovat žádné chyby.

Aplikace také obsahuje v sobě odkaz na dokumentaci od společnosti OMG, kde jsou popsany a vysvětleny činnosti spojené s modelováním v notaci BPMN. Proto je velmi doporučeno si relevantní část dokumentace přečíst v případě kdyby uživatel si cítil, že nerozumí tomu, jak nalezenou chybu by šlo opravit.

7.2 Vývoj

7.2.1 Framework Vue.js

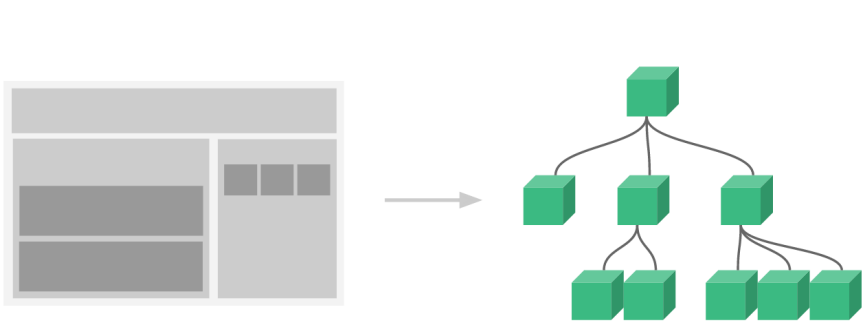
Pro vývoj této aplikace jsem zvolil framework Vue.js a abych s něj mohl aplikace vyvíjet, musel jsem si tento framework předem naučit. Jak již jsem zmiňoval, tento framework má docela podrobnou dokumentaci a na Internetu jsou spousta vhodných kurzů, po absolvování kterých člověk bude mít dobrou představu o tom, jak v tomto frameworku pracovat. V této podkapitole uvedu principy Vue.js, které jsem během implementaci využil.

7.2.2 Komponenty

Stejne jako i u ReactJS, jedním ze základních principů Vue.js je koncept komponent. Komponenty poskytují možnost psát modulární a znovupoužitelný kód pro jakoukoliv aplikaci, postavenou na tomto frameworku. Jako další

přínos bych řekl, že využití komponentů [1] pomáhá jasně definovat architekturu aplikace a usnadňuje práce nad něj. V případě potřeby změny, tato změna se provádí v konkrétním komponentu aplikace. V rámci této práce jsem určitě využil tento princip, však z důvodu, že jedná se o aplikaci, která v zásadě má v sobě jednu fázi nebo jeden případ užití, pro zjednodušení vzhledu aplikace rozhodl jsem si neimplementovat jich příliš hodně.

Pro znázornění na dalším obrázku 7.4 uvádím příklad, jak by mohla vypadat stránka aplikace vyvíjené pomocí Vue.js z technického hlediska.



Obrázek 7.4: Představení aplikace jako strom s hierarchií komponent. [9]

Z tohoto příkladu je snadné vidět, že cokoliv na webové stránce lze představit jako komponent. Například, aplikace může mít komponenty pro hlavičku, postranní lištu (anglicky sidebar), samotnou náplň stránky a tě komponenty mohou mít uvnitř sobě komponenty další.

Struktura jednotlivých komponent je jednoduchá a lze jej rozdělit do třech částí:

- **<template>** - šablona pro jednotlivý komponent. Definuje html prvky, které v sobě komponent bude obsahovat.
- **<script>** - do této části náleží vše funkce spojené s logikou chování komponenty.
- **<style>** - část, ve které je definován vzhled prvků v tomto komponentu.

Po definování komponentu, je potřeba jeho "registrovat". To se dělá dvěma způsoby: buď lokálně nebo globálně. Globální registrace probíhá v hlavním souboru–obecně main.js–a komponent je registrován hned po připojení Vue

[1] V rámci této práce anglicky Component je přeložen do češtiny jako komponent (mužský rod) kvůli protichůdné informaci na Internetu.

a je tato komponenta přístupná z jakéhokoliv dalšího komponentu. Lokální registrace probíhá uvnitř <script> části nějaké z komponent. To znamená, že k registrovanému komponentu bude mít přístup jen ten komponent, který má tento komponent registrovaný ve své <script> části. Po registrace komponent lze jednoduše využit v šablonové části. Jak již jsem zmiňoval, komponenty jsou znovupoužitelné a proto lze v šablonové části inicializovat několik na sobě nezávislých instancí stejné komponenty.

Další důležitou věcí, o které bych chtěl napsat, je předávání dat mezi komponenty. V případě kdybychom chtěli předat data z jedného komponentu do dalšího komponentu je potřeba využití užitečné vlastnosti props, kde definujeme data, které bychom chtěli předat nějakému komponentu. Navíc je potřeba provést operace svázaní (anglicky binding) potřebných dat pomocí v-bind v nadřízeném komponentu. Po provedení těchto dvou operací vývojář bude mít možnost využívat data z nadřízeného komponentu v komponentu podřízeném.

V této práci předávání mezi komponent jsem využil takovým způsobem, že komponent Model v sobě obsahuje atribut errors (je to pole řetezcových hodnot), do kterého postupně se zapisují vše chyby, které byli nalezeny během analýzy modelu. Pak toto pole dat se předává do komponentu ErrorLog, které odpovídá za zobrazení modálního okna s seznamem chyb.

7.3 Knihovna BPMN.io

Pro umožnění vytváření modelů v notaci BPMN jsem využil knihovnu BPMN.io. Je to nástroj vyvíjený společností Camunda a poskytuje v sobě možností prohlížení, zobrazení a kreslení diagramů v notaci BPMN. Tato knihovna je open-source, což znamená, že její zdrojové soubory jsou veřejně přístupné a každý může tento nástroj modifikovat pro vlastní využití. Daná knihovna je napsána v JavaScriptu a umožňuje její využití v jakékoliv aplikace, která je postavena na daném jazyce.

Rozhraní této knihovny poskytuje kreslící plochu (anglicky canvas) a sady nástrojů, pomocí kterých lze samotné procesy na canvasu kreslit. Kromě vykreslovacího nástroje, tato knihovna je taky postavená na webovém modeláří (anglicky web modeler) bpmn-js. Tento webový modelář využívá dvě další knihovny: diagram-js a bpmn-moddle. Diagram-js odpovídá za interakcí s uživatelém, poskytuje služby jako EventBus a odpovídá za rendering a modelování. Bpmn-moddle je využit pro interpretaci a načtení či zapsání modelů z/do formátu XML.

V práce jsem využil poskytnuté služby ElementRegistry, Modeling a Canvas. Z důvodu toho, jak tato knihovna je napsána, je vždy na začátku práce

potřeba načíst si diagram. Z tohoto důvodu po spuštění aplikace uživatel vidí již nakreslený Start Event– je to diagram, který jsem vydefinoval v souboru diagrams.js a který se načítá po spuštění aplikace. Na následujícím obrázku 7.5 bych chtěl uvést kus kódu, který odpovídá za provedení této činnosti.

```
async openDiagram(xml) {
  const self = this;
  try {
    const result = await self.modeler.importXML(xml);
    const {warning} = result;
    console.log(warning);
  } catch(err) {
    console.log(err.message, err.warning);
  }
},
```

Obrázek 7.5: Načtení původního diagramu v aplikaci.

Služba ElementRegistry byla velice užitečná, protože v sobě poskytuje řadu metod, pomocí kterých lze dostat informací o nakreslených prvcích. Mezi nimi bych uvedl tři nejdůležitějších:

- **filter(condition)**. Tato metoda vrací pole prvků, které odpovídají podmínce condition.
- **getAll()**. Tato metoda vrací pole všech prvků v modelu.
- **get(id)**. Tato metoda vrací prvek s identifikátorem id specifikovaným jako argument.

První dvě metody jsem využíval v rámci této práce nejvíc. Metody, které odpovídájí za odhalení chyb vše postaveny na vuyžití těchto poskytnutých knihovnou metod. Vše relevantní funkce, které odpovídají za ověření správností diagramu lze nalézt v souboru checks.js. Jako příklad využití filtrování a služby modeling ukážu příklad funkce, která ověřuje pokud v případě modelování kolaborace je uvažována správná míra detailu. Například, máme účastníka, ze kterého odchází tok zpráv k účastníku číslo dva (ne dovnitř). Avšak účastník číslo jedna má uvnitř svého bazénu tok aktivit, účastník číslo dva ale ne. Toto je porušení principu kolaborace a dokonce, jak již jsem psal v kapitole **"Nejcasteji vyskytující se chyby v modelech notací BPMN"**, nedává žádnou představu o běhu komunikace mezi účastníky. Relevantní kus kódu je ukázán na obrázku 7.6.

```
checkParticipantFlows(elementRegistry, modeling) {
    let ret=[];
    let elems = elementRegistry.filter(el=> el.type==="bpmn:Participant");
    if (elems) {
        elems.forEach(part => {
            if (part.children.length>0 && part.outgoing.length>0) {
                console.log("LOG!");
                let a = part.outgoing;
                a.forEach(flow => {
                    modeling.setColor(flow, colors: {
                        stroke:'red'
                    })
                });
                ret.push(part.businessObject.name);
            }
        })
    }
    return ret;
},
```

Obrázek 7.6: Metoda pro ověření správnosti modelování kolaborace.

Kapitola 8

Uživatelské testování

V této kapitole popíšu uživatelské testování a jeho výsledky včetně závěrů, ke kterým jsem na základě zpětné vazby dospěl.

8.1 Obecný popis testování

Nezbytnou součásti vývoje aplikace bylo i testování. Pro testování jsem využil své kolegy a přatele, avšak kvůli tomu, že práce se psála během leta, uživatelská skupina činila jen čtyři lidí. V ideálu bych chtěl aby moje testovací skupina odpovidala cílové skupině uživatelů, popsané v kapitole **"Požadavky na aplikaci"**, ale bohužel jeden člověk moc o notací BPMN nevěděl, a proto musel jsem pro něho provést krátký úvod do problematiky modelování procesů a samotné notaci BPMN.

Ve všech případech ale, kromě již zmíněného, nechal jsem uživateli si vybrat proces, který chce namodelovat a případně si pohrát s aplikaci a zkusit namodelovat model, chybu ve kterém by aplikace nezvládla nalézt. Každý tester v průměru nad aplikací strávil kolem 30 až 40 minut, co docela odpovídá mému popisu v kapitole **"Požadavky na aplikaci"**. Během modelování testeři jen pracovali sami a případné dotazy týkali se jen panelu nástrojů pro kreslení, který pár testerů nezdál se za intuitivní. Přiznávám se, že když jsem poprvé začal modelovat v bpmn.io, zdálo se mi to taky, ale docela rychle jsem si zvykl na to, jak poskytnutím kreslicím nástrojem lze ovládat.

8.2 Výsledky testování

Testování uživatelem bez předchozí znalosti notace BPMN a procesního modelování dokázalo být méně výnosné, než výsledky testování od zkušenějších uživatelů. Avšak tento kolega mě napověděl, že aplikace může mít problémy s zobrazováním na měnších rozlišeních obrazovek a že bych měl přidat nějaké upozornění ohledně tohoto problému. Navíc od tohoto kolegy jsem dostal zpětnou vazbu, že aplikace by mohla mít přímý odkaz na dokumentace BPMN. Přesně na základě této zpětné vazby jsem přidal do aplikace tlačítko s odkazem na dokumentaci. Však je potřeba mít Internetové připojení aby uživatel na stránky s dokumentací úspěšně dostal.

Zkušenější uživatelé moc problémů s modelovaním neměli, ale důležité je říct, že v případě modelování složitějších procesů aplikace byla schopna odhalit většinu chyb spojených s deadlockem nebo multi merge. V tomto případě se jedná o 80 % úspěšnost (8 modelů z 10). Příčina neodhalení zbývajících 20 % spoléhá v neošetření některých situací v metodě, která odpovídá za detekce tohoto typu chyb. Například, jeden z modelů, který prošel kontrolou (nebyla v něm nalezena chyba) a obsahoval deadlock, byl postaven na principu využití přerušující koncové udalostí (Terminate End Event). Během implementace jsem neuvažoval situace, ve kterých proces obsahuje ten typ prvku a proto nebyla metoda schopna deadlock nalézt.

Navíc další typ chyb se týkal prostě řečeno syntaktické nedávajících smysl vstupů. Jedná z chyb, kterou aplikace schopna nalézt nebyla, se objevila v modelu, ve kterém uživatel přidal do diagramu aktivitu, tu propojil s aktivitou další, a z té další aktivity by šel sekvenční tok zpatky do aktivity původní, co by tvořilo nekoneční smyčku. Tento typ chyb aplikace ani neuvažovala, protože více méně zkušený procesní analytik by takový proces ani namodeloval. Však přijal jsem tuto zpětnou vazbu a je to kontrola, o kterou by šlo aplikace rozšířit.

Testeři, co modelovali kolaboraci, neměli s tímto typem modelů problém a řekl bych, že koncept kolaborací je uživateli dobře pochopen. Ve dvou případech aplikace byla schopna nalézt drobnou chybu (nekorektní využití toku zpráv mezi účastníci v modelu s velkou mírou detailů, tj. v modelu, kde uvnitř účastníku jsou namodelovany procesy) a během testování nebyl nalezen případ chybného kolaboračního modelu, který aplikace nebyla by schopna správně zanalyzovat. Je ale důležité říct, že kdyby uvnitř alespoň jednoho bazénu byl namodelovan proces "neodhalitelného"typu z předchozích bodů, aplikace by stejně chybu odhalit schopna nebyla.

Většina zbývajících chyb, které uživateli dopouštěli a aplikace detekovat schopná byla, byly spojeny s uživatelskou nekompletní znalosti využití grafických prvků notaci BPMN a spojených s tím chyb (například, několik

odchozích toků z jednotlivé aktivity) a by jim šlo předběhnout, pokud by této uživatele měli těsnější vztah s modelováním procesů, jako například měl jsem já během psaní této práce. Pokud bych měl mnohem víc testovacích případů a měl stejné výsledky, podle mě by šlo říct, že aplikace svůj hlavní účel plní a lze jej využít při výuce procesního modelování.

Jako další zpětnou vazbu dostal jsem od uživatelů zaprvé přídaní možnosti nahrat diagrám z externího zdrojů (například, z modeleru bpmn.io, dostupné na `https://demo.bpmn.io/`) a zadruhé poskytnutí možností uložit vytvořený diagram na počítači buď v XML formátu nebo v formátu PNG. S touto zpětnou vazbou většinou souhlasím, ale jako důvod proč jsem vůbec tímto směrem během implementace nemyslel uvádím ten fakt, že podle mého názoru uživatel by měl vytvořit správný model během hodiny (maximálně) a nebyla by potřeba žádný progres ukládat. Navíc uvádím ten fakt, že tato práce je zkušební verzí vytvoření nástroje usnadňujícího odhalení chyb v modelech notací BPMN a tato funkcionalita je už za rozsahem (Scope) práce. Souhlasím ale, že poskytnutí rozhraní pro uložení modelu jako obrázku, by bylo docela vhodným. Protože testování se provádělo v docela pozdní fáze implementaci, bohužel jsem to doho ani nedostal, ale jako dočasné řešení bych navrhnul využití služeb poskytnutých operačním systémem (v případě Windows jsou to tlačítko PrintScreen nebo kombinace Win+Shift+S).

Další věc, kterou jsem dostal jako součást zpětné vazby, bylo podrobnější popsání chyb a pokud je možné upřesnění v tomto popisu kapitol v dokumentace od OMG, do kterých by uživatel měl se kouknout. Původní popis chyb byl o něco kratší než popis chyb, který mám v aplikaci teď a přidání relevantních kapitol v popisu jednotlivých chyb podle mého názoru může usnadnit práce v aplikace a o něco ušetřit čas uživatelům. Poslední věc, kterou jsem dostal od této skupiny testerů bylo označení jména zdrojového elementu v slovním popisu tučným písmenem, ale bohužel kvůli tomu, jak jsem měl implementováno předání těchto dat do příslušného komponentu, nepodařilo se mi to udělat a v slovním popisu (tam, kde to bylo vhodné) přidal jsem uvozovky před jménem jednotlivého elementu, který je odhadovaným zdrojem nalezené chyby.

8.3 Závěry z testování

V závěru bych uvedl, že testování bylo docela přínosnou zkušenosti během vývoje aplikace. Na základě zpětné vazby jsem navrhnul v předchozí kapitole možností rozšíření aplikace v budoucnu a na základě zpětné vazby od vyučujících a studentů by tato aplikace podle mého názoru mohla být zařazena do studia.

Jako nedostatek testování, které jsem provedl, uvedu malou testovací

skupinu a docela pozdní zahájení testování. Přiznávám se ale, že většinou je to moje chyba a do budoucna tuto zkušenost odnesu a snažím tomuto problému předběhnout.

8.4 Alternativy a budoucnost aplikace

Během práce nad touto aplikací jsem si našel několik alternativních řešení, které se také zabývají ověřováním správností modelů a dodržovaním správních postupů. Jako nejpopulárnější uvedu rozšíření knihovny bpmn.io bpmn-js-bpmnlint [29]. Tato knihovna je vyvíjená v rámci projektu BPMN.io a poskytuje základní a pokročilejší analýzu modelovaných procesů. V rámci této práce rozhodl jsem si tuto knihovnu nevyužívat i kdybych stejně měl většinu potřebných metod implementovat.

Osobně si myslím, že moje aplikace bude pro studenty užitečnější, protože analýzuje i chyby, které v funkcionálu bpmnlintu v době psaní této práci nejsou. Ale z důvodu nedostatečně velké zkušenosti s front-end vývojem také myslím, že vzhled a některý funkcionál moje aplikace by stejne šlo zlepšit pro ještě lepší uživatelskou zkušenost.

Kromě možností rozšíření, které jsem již popsál v výsledcích testování jako další možnost rozšíření této aplikace vidím její sjednocení s aplikací, kterou vyvíjel kolega Ing. Jan Polan [25], na práce kterého navazuju, a s aplikací kolegy Zakhutského [28], který stejně jako já navazoval na práce kolegy Polana. Stejně jako bpmnlint aplikace by šlo rozšiřovat přidáním nových pravidel a kontrol podle potřeb uživatelů.

Pro zařazení aplikace do výuky v nejbližší budoucnosti by bylo vhodně prodiskutovat požadavky s vedoucím práce a provést další testování s větším počtem účastníků pro odchycení dalších nesrovnalostí, které nebyly identifikovány během původního testování. Na základě výsledků z tohoto testování by aplikace šlo doplnit o další pravidla (podle potřeby) a případně obnovit stávající algoritmy pro zvýšení efektivity ověřování správností modelů.

8.5 Závěr

V této kapitole jsem popsal proces testování moje aplikace včetně jejího výsledku. V těch výsledcích jsem provedl stručnou analýzu těchto výsledků včetně příčin vzniku situací, kdy aplikace nebyla schopna odhalit 100 % všech chyb. Na konci jsem zmínil alternativní řešení od společnosti Camunda, které také se zabývá validací procesů a popsal jsem možné rozšíření aplikaci a její

budoucnost. V kapitole další popíšu moje vlastní ocenění funkčností aplikací a využitích algoritmů v něj.

Kapitola 9

Shrnutí stavu aplikace

V této kapitole bych chtěl provést shrnutí stavu celé aplikace, a to včetně mého vlastního vyhodnocení její funkčnosti na základě výsledků testování, popsaných v kapitole **“Uživatelské testování”**.

Podle mého názoru aplikace své základní požadavky splňuje, což bych podpořil výsledky testování. I když během testování byli nalezeny situace, které aplikace detekovat jako chybné schopna nebyla, toto je způsobeno tím, že velmi těžko se stanoví úroveň znalostí průměrného uživatele z cílové skupiny. Méně zkušený uživatel může dopustit chybu, kterou jsem ani neuvažoval za chybu, kterou by někdo z této cílové skupiny vůbec dopustil.

Však vše chyby, které byli popsány v kapitole **“Požadavky na aplikaci”**, aplikace detekovat s velkou pravděpodobností schopna, což se ukázalo v testovací fáze. Z tohoto pohledu by podle mého názoru šlo říct, že aplikace je “hotová”.

Funkčnost stávajících metod na základě existujících výsledků testování hodnotím jako dobrou. Méně složité funkce jako kontrola dodržování stejné úrovně detailu nebo kontrola jmen počátečních a koncových elementů fungují úplně správně a jich potřeba měnit asi nevznikne. Co se týká funkcí, které odpovídají za detekce deadlocků a multi merge problémů, podle mého názoru využité mnou algoritmy by určitě šlo zlepšit. Určitě nepovažují stávající algoritmus v těchto funkcích za neporušitelný. Momentálně ale nemám konkrétní návrh, se kterým bych šel implementovat až teď. Minimálně by určitě šlo této metody rozšířit o zahrnutí chyb, které nebyl stávající algoritmus schopen detekovat (viz popis výsledků testování v kapitole “Uživatelské testování). Tento algoritmus ale podle mého názoru by mohl stačit pro uživatele, které jsou cílovou skupinou této aplikaci.

Jako závěr bych řekl, že v případě potřeby ošetření většího počtů chyb,

je aplikace v tomto směru dobře rozšiřitelná, což je dobrou vlastností pro budoucnost této aplikaci, a by mohla jako důsledek být zařazena do výuky procesního řízení.

Kapitola 10

Závěr

V rámci této práce bylo vydefinovano několik cílů.

První cíl práce byl definovat proces a procesní řízení. Za tímto účelem jsem se seznámil a popsal jsem v první kapitole proces a spojené s ním vlastnosti včetně životního cyklu. Následně v této kapitole popsal jsem oblast procesního řízení, jeho výhody a nevýhody. V kapitole další krátce jsem popsal problematiku modelování procesů z důvodu, že jako druhý cíl měl jsem seznámit se s notaci BPMN 2.0, která zabývá se modelováním procesů.

Proto třetí kapitola obsahuje popis notaci BPMN. Začal jsem krátkým popisem historie vývoje této notace a pokračoval jsem popisem základních grafických prvků. Následně jsem se věnoval typů modelů, provedl jsem porovnání této notace s další široce využívanou notaci UML a krátce jsem popsal zodpovědnosti procesního analytika. V závěru jsem uvedl příklad mnou namapovaného procesu v této notace jako příklad využití notaci BPMN v reálním životě.

Třetí cíl této práce bylo provedení rešerše nejčastěji vyskytujících se chyb procesních analytiků během modelování. Proto ve čtvrté kapitole se zabývám definováním jednotlivých chyb a jejích analýzou, která je potřebnou části pro splnění dalšího cíle této práce.

Další cíl byl navrhnout a implementovat aplikace, která by umožnila uživateli vytvořit diagram obsahující základní grafické prvky v notaci BPMN, a provést kontrolu její správnosti na základě výsledků provedené analýzy v předchozím bodě. Pro tento účel v kapitole číslo pět jsem popsal funkční a nefunkční požadavky na aplikaci, která je výstupem této práce. V kapitole číslo šest jsem uvedl seznam typu chyb, které aplikace schopna detekovat a krátce jsem zmínil technická omezení, kvůli kterým některé chyby z předchozí analýzy se nedálo implementovat. Tento popis je následován rozborem logiky

chování jednotlivých metod, odpovídajících za detekce příslušného typu chyb v aplikace. V kapitole číslo sedm jsem popsal proces práce s aplikací a poznámky z vývoje včetně rozboru využitých technologií a knihoven s ukázkami z kódu aplikace.

Posledním cílem této práce bylo ověření funkčnosti aplikace na skupině uživatelů a proto kapitola číslo osm je věnována popisu procesu testování a výsledků, ke kterým jsem po provedení této činnosti dospěl.

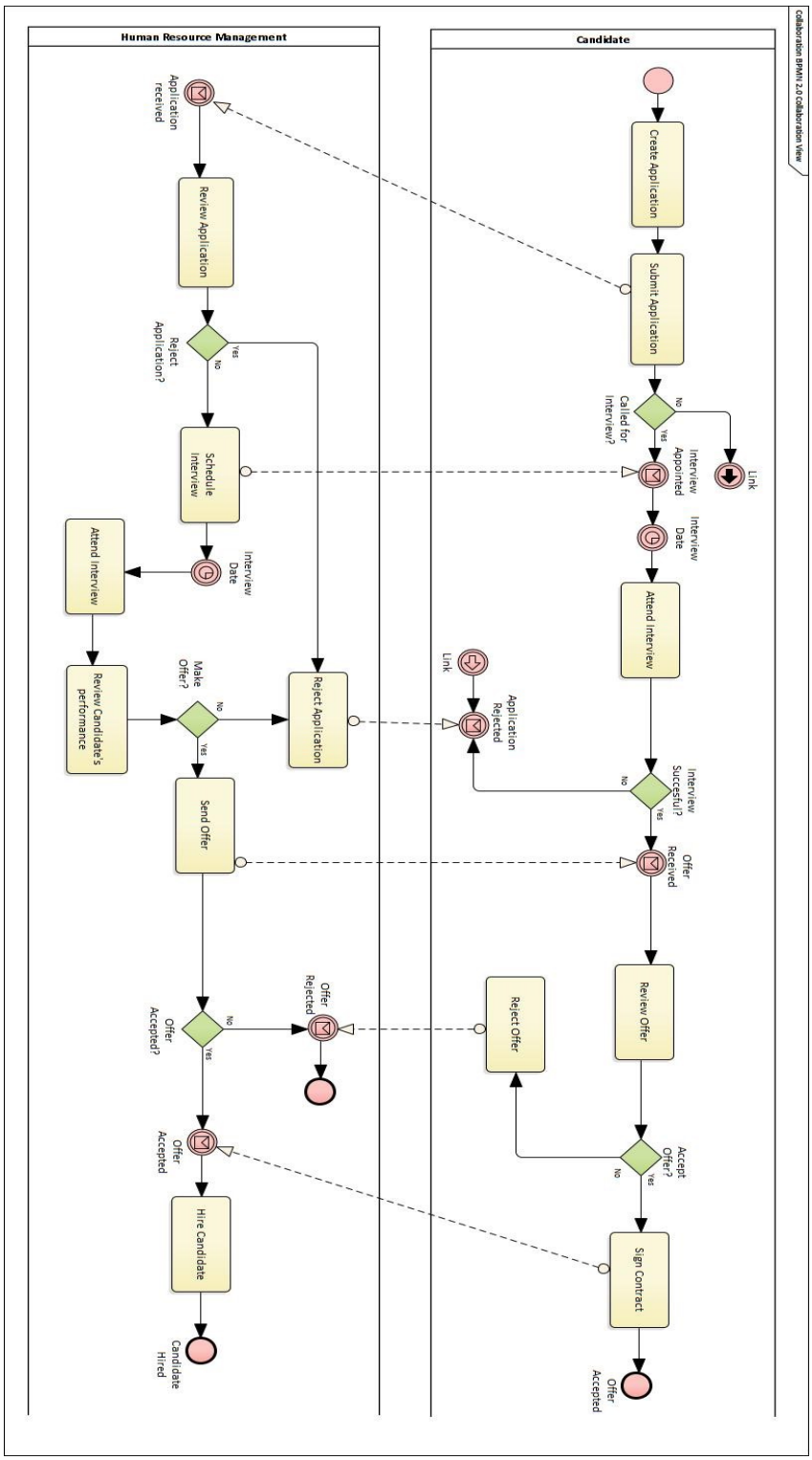
Výsledkem práce je funkční aplikace, která umožňuje uživateli provést kontrolu svých znalostí modelování v notaci BPMN prostřednictvím analýzy nakresleného jím modelu na předmět chyb, které byly definovány a popsány v kapitole číslo šest. V případě nalezení chyby je uživatel upozorněn na něj prostřednictvím barevného označení v modelu a také mu představen slovní popis nalezené chyby pro snadnější provedení opravy.

V závislosti od dalšího a rozsahlejšího testování a zpětné vazby od vedoucího této práce by tato aplikace mohla být zařazena do studia jako například příručka pro studenty, kteři modelují procesy v rámci takových předmětů jako "Procesní řízení". Chtěl bych doufat, že tato aplikace bude pro nich užitečná a pomůže jim při studiu.

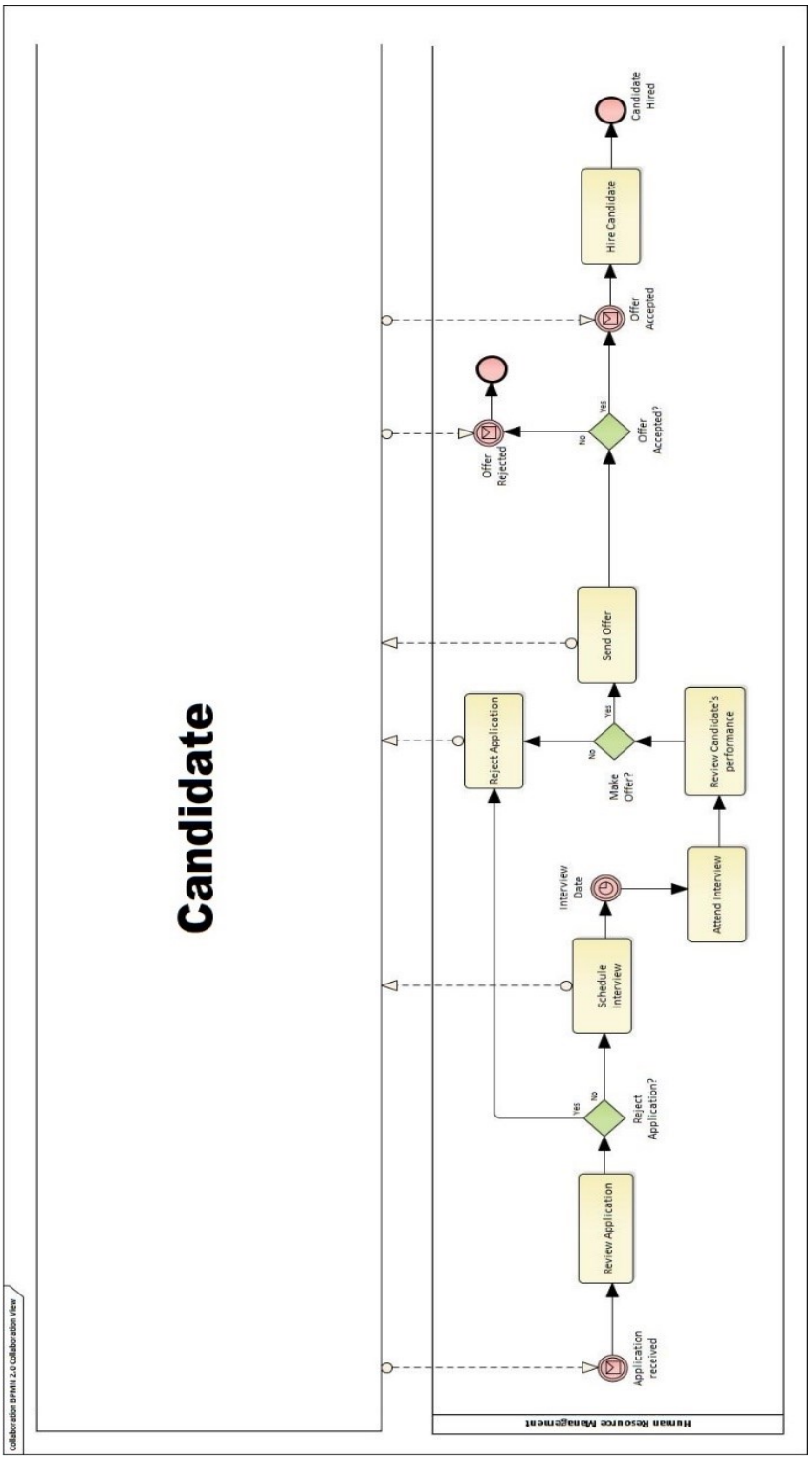
Jako hlavní výsledek této práce bych uvedl těsnější seznámení s notaci BPMN, obdržení nových znalostí z problematiky procesního řízení, procesního modelování a získání zkušenosti vývoje front-endové aplikace za využitím neznámého frameworku. Chtěl bych věřit, že mnou získané teoretické znalosti mně budou v budoucnu a v praxi přínosné.

Příloha A

Proces přijetí zaměstnance



Obrázek A.1: Proces přijetí zaměstnance



Obrázek A.2: Proces přijetí zaměstnance zjednodušený

Příloha B

Literatura

[1] Produkční proces (production process). https://managementmania.com/cs/produkcni-proces, navštíveno 10.5.2020.

[2] OMG. Bpmn 2.0 specification. https://www.omg.org/spec/BPMN/2.0/PDF/, navštíveno 6.6.2020.

[3] Bpmn private business process. https://www.smartdraw.com/bpmn/examples/bpmn-private-business-process/, navštíveno 22.6.2020.

[4] Bpmn orchestration vs choreography vs collaboration. https://www.visual-paradigm.com/guide/bpmn/bpmn-orchestration-vs-choreography-vs-collaboration/, navštíveno 21.6.2020.

[5] H. Leopold, J. Mendling, and O. Günther. Learning from Quality Issues of BPMN Models from Industry. *IEEE Software*, 33:26–33, July 2016.

[6] Bpmn deadlock. http://tynerblain.com/blog/2006/09/20/bpmn-deadlock/, navštíveno 21.6.2020.

[7] Bpmn modeling guidelines. https://www.modeling-guidelines.org/guidelines/absence-of-multi-merges/, navštíveno 21.6.2020.

[8] Maude specification and verification of bpmn processes. http://maude.lcc.uma.es/BPMN-P/, navštíveno 21.6.2020.

[9] Components Basics. https://vuejs.org/v2/guide/components.html, navštíveno 14.7.2020.

[10] e-iso slovník. https://www.eiso.cz/informacni-servis/eiso-slovnik/, navštíveno 18.5.2020.

[11] Šmída Filip. *Zavádění a rozvoj procesního řízení ve firmě*. Praha: GradaPublishing, 2007. ISBN 978-80-247-1679-4.

[12] CZM. Procesní řízení. http://blog.czm-cvut.cz/2015/11/13/procesni-rizeni/, navštíveno 15.5.2020.

[13] Řízení procesů (process management). https://managementmania.com/cs/rizeni-procesu.

[14] Josef Basl, Miroslav Tůma, and Vit Glasl. *Modelování a optimalizace podnikových procesů.* Plzeň: Západočeská univerzita, Strojní fakulta, 2002. ISBN 80-7082-936-2.

[15] ITIL. Procesní řízení (online). https://web.archive.org/web/20100802053030/http://www.itil.cz/index.php?id=914, navštíveno 27.4.2020.

[16] V. Řepa. *Procesně řízená organizace.* Praha: Grada Publishing.

[17] P. Lukasík and J. Procházka. *Procesní řízení: Text pro distanční studium.* Ostrava: Ostravská univerzita, 2013. Dostupne z https://web.archive.org/web/20131228075751/http://www1.osu.cz/~prochazka/rpri/skripta_ProcesniRizeni.pdf.

[18] A. Höffer. *Procesní přístup k řízení organizací a jeho podpora v ERP systémech.* Praha: VŠE, 2006.

[19] Modelování podnikových procesů. https://www.altaxo.cz/provoz-firmy/management/rizeni-podniku/modelovani-podnikovych-procesu, navštíveno 21.6.2020.

[20] W. Bandara, G. G. Gable, and M. Rosemann. *Factors and Measures of Business Process Modelling: Model Building Through a Multiple Case Study.*

[21] Business process modeling: Definition, benefits and techniques. https://tallyfy.com/business-process-modeling/, navštíveno 21.6.2020.

[22] Business Process Modeling. https://studme.org/62398/menedzhment/modelirovanie_biznes-protsessov, navštíveno 1.7.2020.

[23] A. Correia. Elements of style of bpmn language. 2000.

[24] Difference between uml and bpmn. http://www.differencebetween.net/technology/difference-between-uml-and-bpmn/, navštíveno 18.5.2020.

[25] J. Polan. *Virtuální příprava procesních analytiků.* Praha, 2019. Diplomová práce. ČVUT. Fakulta elektrotechnická. Vedoucí práce Pavel Náplava.

[26] J. Fiala and J. Ministr. *Průvodce analýzou a modelováním procesů. 1. vyd.* Ostrava : VŠB -Technická univerzita, 2003. ISBN 80-248-0500-6.

[27] M. Muehlen and J.Recker. *How Much Language Is Enough? Theoretical andPractical Use of the Business Process Modeling Notation*. Springer-Verlag, 2008.

[28] V. Zakhutskyi. *Simulátor modelování procesů v notaci BPMN*. Praha, 2020. Bakalářská práce. ČVUT. Fakulta elektrotechnická. Vedoucí práce Pavel Náplava.

[29] bpmn-js-bpmnlint. `https://github.com/bpmn-io/bpmn-js-bpmnlint`, navštíveno 17.7.2020.