ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zakhutskyi** Jméno: **Viacheslav** Osobní číslo: **461283**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Simulátor modelování procesů v notaci BPMN

Název bakalářské práce anglicky:

Simulator of BPMN process modelling

Pokyny pro vypracování:

Navažte na diplomovou práci Jana Polana, který ověřil možnosti tvorby BPMN modelů a jejich následné testování prostřednictvím nástroje Camunda. Vytvořte jednoduchý simulátor, který ověří schopnost uživatele převést slovní popis procesu do podoby modelu v notaci BPMN. Postupujte následovně:
1) Definujte pojmy proces, procesní řízení, procesní analytik.
2) Seznamte se s notací BPMN (verze 2.0).
3) Společně s vedoucím práce navrhněte a popište několik reálných situací, které představují proces a je možné je převést do modelu v notaci BPMN.
4) Navrhněte a implementujte aplikaci, která formou simulace ověří schopnost uživatele identifikovat ve slovním popisu správné elementy notace BPMN, propojit je do modelu, který dodržuje specifikaci notace a reflektuje slovní popis. Pokud nebude vytvořený model odpovídat slovnímu popisu nebo specifikaci notace BPMN, bude uživatel upozorněn na chyby a chybějící části.
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] Bjørn Andersen, Business Process Improvement Toolbox, 2007
[3] Hammer, M. L. W. Hershman, Rychleji, levněji, lépe, Devět faktorů účinné transformace podnikových procesů, Management Press, 2013
[4] Bruce Silver, BPMN Method and Style, 2nd Edition, with BPMN Implementer's Guide, Cody-Cassidy Press, 2011

Jméno a pracoviště vedoucí(ho) bakalářské práce:

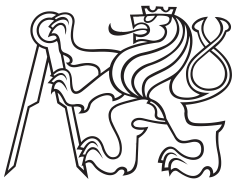
Ing. Pavel Náplava, Ph.D., katedra ekonomiky, manažerství a humanitních věd FEL

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

Datum zadání bakalářské práce: **14.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

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)



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Simulátor modelování procesů v notaci BPMN

Viacheslav Zakhutskyi

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

Poděkování

Rád bych poděkoval vedoucímu práce, Ing. Pavlovi Náplavovi za cenné rady a připomínky při tvorbě práce. Všem účastníkům testování za jejich trpělivost a ochotu se testování zúčastnit. Také bych chtěl poděkovat své rodině, přítelkyni a přátelům za jejich neutuchající podporu při studiu.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 22. května 2020

Abstrakt

Tato bakalářská práce se zabývá procesním řízením a modelováním procesů v notaci BPMN. Základem je seznámení se s pojmem procesu a s aspekty procesního řízení. Dále je definovaná oblast modelování procesu a následně je zmíněná notace BPMN. Součástí práce je popis činnosti a nezbytných dovedností procesního analytika a souhrn možností jeho vzdělávání. V další části práce je představen návrh a realizace aplikace, která umí ověřovat schopnost uživatele porozumět slovnímu popisu procesu a následně ho převádět do podoby modelu v notaci BPMN. Poslední části práce obsahuje vyhodnocení jejích výsledků pomocí testování a krátký popis jejího možného budoucího rozvoje. Výslednou aplikaci lze využít při výuce procesního řízení.

Klíčová slova: Procesní řízení, modelování procesů, simulátor, BPMN, procesní analytik

Abstract

This bachelor thesis is dedicated to the process management and process modeling in BPMN notation. The basis is an introduction to the process concept and aspects of the process management. Further a process modeling area is defined, then BPMN notation is mentioned. Part of the thesis is also a description of the activities and necessary skills of the process analyst and a summary of the ways of his education. The next part presents the concept and implementation of an application that can test a user's ability to understand a verbal description of the process and then to transform it into a model in BPMN notation. The last part of the thesis contains the results of testing the application and a brief description of its possible future development. The resulting application can be used in the process management training.

Keywords: Process management, process modeling, simulator, BPMN, process analyst

Obsah

Obrázky

Tabulky

Úvod

Tato práce se zabývá problematikou procesního řízení a její výukou, modelováním procesů pomocí využití notace BPMN a s tím spojenými činnostmi.

Motivace

Primární motivací k tvorbě této práce byla snaha vytvořit užitečný nástroj, který bude v budoucnosti využit jako doplnění výuky zájemců o procesní řízení a který zvýší kvalitu a zajímavost výuky. Osobní motivací bylo v první řadě dosáhnutí definovaných cílů, ve druhé řadě naučit se nové technologie a udělat takovou práci, na kterou bude možné dále navázat.

Cíle práce

Prvním cílem je seznámení se s oblastí procesního řízení a následným modelování procesů. Druhým cílem je prostudování notace BPMN a určení jejích klíčových aspektů. Třetím cílem je prostudování profese procesního analytika a již existujících způsobů vzdělávání v této oblasti. Čtvrtým cílem je na základě provedené rešerše navrhnout a implementovat simulátor, který ověří schopnost uživatele převést slovní popis procesu do podoby modelu v notaci BPMN.

Struktura práce

Práci lze rozdělit do dvou klíčových částí, kde první část je věnovaná teoretické části a druhá praktické. Každá z těchto částí obsahuje pět jednotlivých kapitol. V první kapitole popisuji oblast procesního řízení, definuji základní pojmy používané v této oblasti a zabývám se základními aspekty procesu a procesního řízení. Ve druhé kapitole se zaměřuji na modelování procesů a jeho součásti. Následující kapitola je věnovaná modelovací notaci BPMN. Ve čtvrté kapitole se dotýkám profese procesního analytika a specifikuji jeho činnosti a dovednosti. V páté kapitole rozebírám existující způsoby vzdělávání procesního analytika. V šesté kapitole navazuji na celkovou teoretickou část a definuji požadavky na aplikaci. Sedmá kapitola obsahuje popis logiky a

uživatelského rozhraní aplikace. V další kapitole se věnuji vnitřním datům aplikace a důvodům, podle kterých jsem je zvolil. Devátá kapitola obsahuje objasnění podstatných a důležitých věcí, které se týkají vývoje aplikace. V desáté kapitole se věnuji testování aplikace a krátkému popisu její budoucnosti.

Kapitola 1

Procesní řízení a proces

V první kapitole se zabývám procesním řízením a procesem. V úvodu se zaměřuji na krátký popis historie procesního řízení. Dále představuji pojem proces. Formuluji definice, vlastnosti, klasifikaci a životní cyklus procesu. Potom se zaměřuji na základní hlediska procesního řízení, kde představuji definici, principy, přínosy a omezení procesního řízení. Jako závěr formuluji shrnutí této kapitoly.

1.1 Úvod do procesního řízení

Procesní přístup byl jako koncept holistického řízení vytvořen na přelomu 80. a 90. let 20. století. Významnou a důležitou událostí ve světě řízení bylo vydání knihy Hammer a Champi „Reengineering the Corporation: A Manifesto for Business Revolution“ v roce 1993. Ačkoli metodika reengineeringu navržená autory [9] nefungovala, přesto hrála významnou roli v popularizaci procesního přístupu.

Do konce 20. století se kvalita produktů většiny výrobců ustálila a v boji o pozici na trhu se důraz přesunul směrem k efektivitě a účinnosti, která se stala jednou z hlavních konkurenčních výhod.

Celým problémem však bylo, že obvyklý funkční hierarchický systém organizace a řízení neměl na zlepšování efektivity dobrý vliv. Jasně byl vyžadován jiný přístup, kterým se stalo procesní řízení.

„Záměrem této změny je zvýšení flexibility firem ve vztahu ke schopnosti přizpůsobit se měnícím podmínkám. Procesní řízení je postaveno na uplatňování integrace činností při řízení firem. Oproti funkčnímu managementu dochází k opačnému postupu, a to sjednocení dílčích operací do podnikových procesů, které jsou ovládány procesními týmy a řízeny jejich vlastníky. Hlavním kritériem měření výkonnosti jednotlivých procesů je vytvořená hodnota pro zákazníka.“ [10]

1.2 Proces

1.2.1 Definice procesu

Základ procesního řízení tvoří proces, proto je důležité uvést jeho formální definici. Existuje několik různých definic pojmu proces a o žádné z nich nelze říct, že je ta nejpřesnější. Pro širší přehled uvádím vícero definic tohoto pojmu.

Definice procesu podle normy ISO 9001 je následující:

„Proces je soubor vzájemně souvisejících anebo vzájemně působících činností, který přeměňuje vstupy na výstupy.“ [11]

Tato norma nabízí příliš stručnou definici, která nepočítá třeba s různými externími zdroji, které mohou působit na proces. Podrobnější definicí je například:

„Proces je organizovaná skupina vzájemně souvisejících činností a/nebo subprocesů, které procházejí jedním nebo více organizačními útvary či jednou (podnikový proces) nebo více spolupracujícími organizacemi (mezipodnikový proces), které spotřebovávají materiální, lidské, finanční a informační vstupy a jejichž výstupem je produkt, který má hodnotu pro externího nebo interního zákazníka.“ [12]

Obecně je tedy proces opakující se souhrn jedné nebo více souvisejících operací nebo postupů, které kolektivně provádějí konkrétní cíl produkční činnosti, obvykle prováděný v předem stanovené organizační struktuře, která představuje vztahy mezi účastníky.

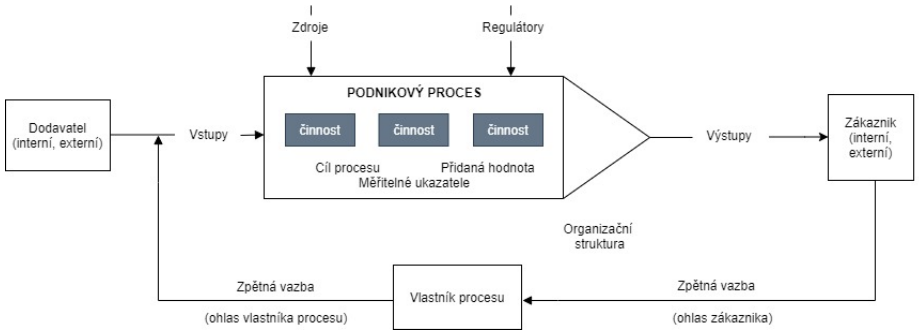
1.2.2 Vlastnosti procesu

V každém procesu existují stejné vlastnosti, mezi které ve shodě s [13] patří:

- Cíl – to, kam má proces směřovat
- Metriky – jak se daří jednotlivé cíle naplňovat
- Vlastník – osoba, která odpovídá za výsledek a účinnost procesu
- Zákazník – osoba, které je určen výstup procesu
- Vstup – požadavek, materiály nebo informace, které iniciují začátek procesu, a je k nim procesem přidaná hodnota
- Zdroje – materiální nebo informační objekty, pomocí nichž jsou vstupy přeměněné na výstupy
- Výstup – materiální a informační objekty, které jsou výsledkem procesu a oceněny zákazníkem

- Rizika – činnosti, které ohrožují nebo negativně působí buď na průběh, nebo výsledek procesu
- Regulátory – trvalá pravidla, která mohou do jednotlivých procesů vstupovat
- Činnosti – sled měřitelných pracovních úkolů vykonaných v rámci jedné organizace

Definiční model procesu včetně jeho vlastností ilustruje obrázek 1.1:



Obrázek 1.1: Model procesu včetně jeho charakteristik [1]

1.2.3 Rozdělení procesů

Existuje řada přístupů k rozdělení procesů. Tradičně se procesy člení na tři hlavní skupiny podle jejich působení na hodnotu výstupu: **řídící**, **hlavní** a **vedlejší**. Podle [14] lze k těmto skupinám přidat ještě dvě, a to **podpůrné** a **sdílené**, kde:

- **Řídící** procesy jsou zodpovědné za vytvoření účinného systému řízení, který splňuje cíle organizace.
- **Hlavní** procesy mají za úkol dosáhnout plnění cílů organizace.
- **Vedlejší** procesy se nepodílejí přímo na hlavní činnosti organizace, ale bez nich nelze provádět hlavní procesy.
- **Podpůrné** procesy se zabývají podporováním hlavních procesů a zajištěním možnosti jejich vykonávání.
- **Sdílené** procesy jsou určené k zajištění správné funkčnosti všech procesů v rámci organizace.

Další možnou klasifikací procesu je rozdělení podle standardu ISO 9001:2009, které rozděluje procesy na základě jejich zacílení na zástupce [11]: **řídící**, **přípravy zdrojů**, **realizace produktů** a **dalšího rozvoje**.

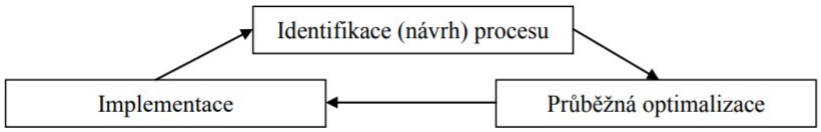
Za zmínku také stojí myšlenka [15], že dělení na **hlavní** a **podpůrné** procesy může někdy působit zavádějícím dojmem. Stává se to z toho důvodu, že hlavní a podpůrné procesy jsou velmi podobné a velmi často je složité je rozlišit.

1.2.4 Životní cyklus procesu

Kromě definice, charakteristik a klasifikace lze proces popsat pomocí životního cyklu. Životní cyklus procesu je posloupnost fází, které určují dynamiku implementace a vývoje procesu.

V rámci procesního přístupu je životní cyklus procesu posuzován jako řízení vývoje, implementace, údržby a zlepšování procesu.

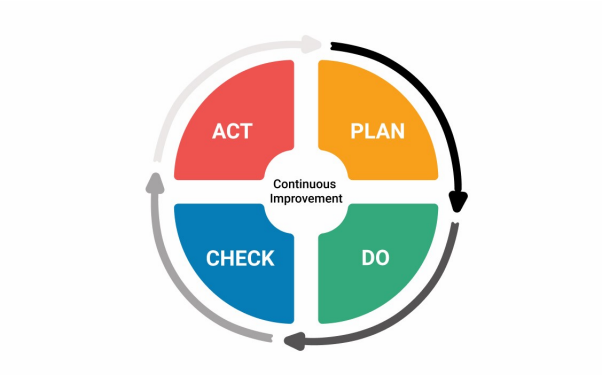
Podle [2] lze životný cyklus procesů rozdělit do tří hlavních etap: identifikace nebo návrh procesů, průběžná optimalizace a implementace. Autor uvádí, že procesy je zapotřebí neustále revidovat a optimalizovat. Proto je potřeba zrušit neefektivní činnosti a případně nahradit novými. Tento postup je znázorněn na obrázku 1.2.



Obrázek 1.2: Životní cyklus procesů podle [2]

Velmi známou metodou řízení procesů je **Demingův cyklus**, jehož hlavní myšlenkou je neustálá postupná optimalizace kvality procesu přes opakované provádění čtyř základních činnosti (viz obrázek 1.3), které jsou:

- P (Plan) – analýza aktuálního stavu a plánování zlepšení
- D (Do) – realizace a testování plánu
- C (Check) – ověření a hodnocení výsledku
- A (Act) – implementace a případné optimalizování



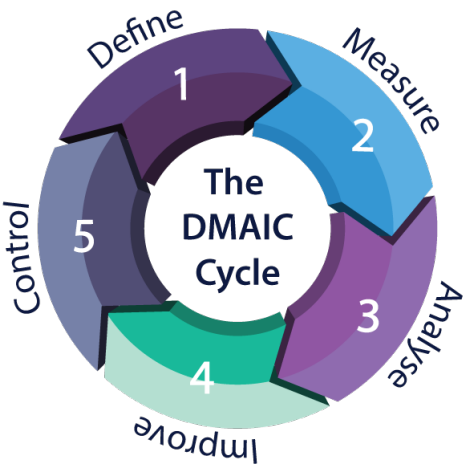
Obrázek 1.3: Demingův cyklus [3]

Existuje taky přístup **DMAIC** (viz obrázek 1.4), který se používá v metodě Six Sigma. Definice metody Six Sigma je následující:

„Metoda Six Sigma je flexibilní a úplný systém dosahování, udržování a maximalizace obchodního úspěchu. Je založena na porozumění a očekávání zákazníků, správném používání dat, faktů a na detailní statistické analýze a na základě pečlivého přístupu k řízení, zlepšování a vytváření nových výrobních, obchodních a obslužných procesů.“ [16]

Tento přístup staví na PDCA cyklu a je jeho zdokonalenou verzí. V rámci DMAIC jsou definované následující součásti:

- D (Define) – definování příležitosti ke zlepšení
- M (Measure) – měření efektivity existujících procesů
- A (Analyze) – analýza informací a aktuálního stavu
- I (Improve) – zlepšení na základě předchozí analýzy
- C (Control) – zavedení přijatých změn a jejich následná kontrola



Obrázek 1.4: Přístup DMAIC [4]

1.3 Základní aspekty procesního řízení

Procesní řízení neboli Business Process Management (zkráceně BPM), se zabývá způsobem řízení organizace, při němž jsou všechny činnosti organizace představené jako sítě vzájemně propojených podnikových procesů, nikoliv jako kombinace různých samostatně působících funkcí. Podle [13] lze procesní řízení charakterizovat následně:

„Základní charakteristikou procesního řízení je schopnost pružně reagovat na všemožné požadavky zákazníků a jejich plnění. Procesní přístup nám tedy

poskytuje elastický přechod od jednoho požadavku zákazníka ke druhému.“

Mezi cíle procesního řízení patří [13]:
„Cílem procesního řízení je rozvíjet a optimalizovat chod organizace tak, aby efektivně a účelně a hospodárně reagovala na požadavky zákazníka
A) způsobem, který:

- Definuje pracovní postup (proces) jako ucelený sled činností napříč organizací;
- Pro každý proces definuje jeho vstupy, výstupy a zdroje;
- Definuje osobní zodpovědnost za proces i za každou činnost;
- Nastavuje systém měření výkonnosti procesu;
- Sleduje a vyhodnocuje každý proces;

B) tak, aby:

- Byla dodržována kvalita výsledků procesů daná měřenými ukazateli a jejich parametry;
- Byly optimálně využívány dostupné zdroje;
- Byla průběžně zvyšována výkonnost organizace dle předem známých a měřených ukazatelů.”

1.3.1 Definice procesního řízení

V předchozích kapitolách jsem se zabýval procesem, bez jehož zmínění, bych se nemohl posunout k definici procesního řízení. Stejně jako v podkapitole „**Definice Procesu**“ bych uvedl několik různých definic pro specifikaci popisované oblasti z různých úhlů.

„*Procesní řízení (management) představuje systémy, postupy, metody a nástroje trvalého zajištění maximální výkonnosti a neustálého zlepšování podnikových procesů i mezipodnikových procesů, které vycházejí z jasně definované strategie organizace a jejichž cílem je naplnit stanovené strategické cíle.*“ [12]

„*Business Process Management (BPM) je soubor metod, nástrojů a technologií používaných pro navrhování, schvalování, analyzování a řízení provozu podnikových procesů. BPM je procesně orientovaný přístup pro zlepšení výkonu, který spojuje informační technologie s procesy a metodikami řízení. BPM je spolupráce mezi obchodníky a informačními technology podporující efektivní, agilní a transparentní obchodní procesy. BPM zahrnuje lidi, systémy, funkce, podniky, zákazníky, dodavatele a partnery.*” [17]

1.3.2 Principy procesního řízení

Podle [18] existuje deset důležitých principů procesního řízení, které je nutné dodržovat ke správnému uplatnění procesního řízení. Tyto principy se „*mohou vázat na práci, proces a podnik jako celek*“:

Práce

- Integrace a komprese prací – sjednocení sledu prací do jednoho logického celku za účelem zvýšení přidané hodnoty.
- Delinearizace prací – výkon práce je prováděn ve přirozeném sledu.
- Nejvýhodnější místo realizace prací – práce je vykonávána na nejvhodnějším místě.

Proces

- Uplatnění týmové práce – procesy jsou zajišťovány autonomními týmy s dostatečnými pravomocemi tak, aby jejich motivace mohla být těsně svázána s naplněním požadavků zákazníka.
- Procesně zaměřená motivace – motivace je přímo svázána s výsledkem procesu.
- Odpovědnost za proces – proces musí mít svého vlastníka, který je za něj zodpovědný.
- Variantní pojetí procesu – proces může mít několik variantních provedení, jejichž volba závisí na velkém množství parametrů.
- 3S (samořízení, samokontrola, samoorganizace) – princip naprosté autonomie týmu.

Podnik

- Pružná autonomie procesních týmů – struktura týmu musí umožňovat přizpůsobení se novým požadavkům.
- Znalostní a informační bezbariérovost – odstranění znalostních a informačních bariér pomocí sdílených databází a centralizovaných informačních zdrojů.

1.3.3 Přínosy procesního řízení

Během studia související literatury jsem narazil na řadu různých možností specifikace přínosů procesního řízení. Třeba [13] uvádí, že přínosy procesního řízení se projevují v různých oblastech organizace a rozděluje podle nich

přínosy. K těmto oblastem patří: řízení organizace, personální zdroje organizace, finanční plánování, logistika, informační technologie a provoz odborných útvarů.

Pro obecnou představu přínosů procesního řízení je podle mého osobního názoru důležité mít přehled klíčových bodů, které přináší zavedení procesního řízení. Mezi tyto klíčové body patří [19]:

- *Zlepšení monitoringu* – usnadňuje monitorování aktuálního stavu organizace, efektivity a kvality práce, a hodnocení stanovených strategických cílů. Kromě toho zjednodušuje analýzu případných příčin, které negativně působí na vytyčené cíle.
- *Zjednodušení řízení změn* – umožňuje organizaci být flexibilní a rychle reagovat na jakékoli změny způsobené vnějšími příčinami.
- *Motivační nástroj* – díky zajištění pracovníků dostatečnými pravomocemi lze zvýšit jejich efektivitu a snadněji propojit stanovené cíle s odměnami.
- *Zpřehlednění potřebných schopností pracovníků* – jasný a přehledný systém vzájemných vazeb uvnitř organizace umožňuje konkrétně definovat potřebné kompetence pro pracovníka, díky čemuž organizace působí soudržně a konzistentně.
- *Logistika* – umožňuje odhalit a odstranit úzká místa v procesech zásobování, optimalizovat logistické procesy a vytvořit nová pravidla v oblasti výrobních podniků.

1.3.4 Omezení a nevýhody procesního řízení

Jednoznačně formulovat omezení procesního řízení není triviální činnosti. Nevýhody se mohou značně lišit podle úhlu, ze kterého se díváme na procesní řízení. Tak třeba nevýhodou z pohledu zaměstnanců je snižování počtu pracovních míst, což zároveň může být efektivním opatřením pro účel optimalizace procesu firmy.

Ve shodě s [20] se dá omezení definovat podle činnosti konkrétní firmy. To znamená, že nevýhody se budou vždycky lišit v závislosti na strategických cílech a procesu organizace. Mezi všeobecná omezení a nevýhody patří:

- Nedostatek motivaci ke změnám, které jsou nutné k plnění cílů
- Špatně definované cíle prováděných změn
- Chybějící stimulace zájmu zaměstnanců vlastníkem procesu
- Nedostatečné znalosti a schopnosti z oblasti procesního řízení
- Obavy zaměstnanců z propouštění
- Podcenění role vlastníků procesu
- Nízká míra zapojení pracovníků

1.4 Závěr první kapitoly

V první kapitole jsem se zaměřil na pojmy proces a procesní řízení, objasnil jsem jejich termíny a probral jsem důležité aspekty prostředí procesního řízení. Na začátku jsem se zabýval úvodem do procesního řízení a krátkým náhledem do jeho historie. Potom jsem uvedl definice procesu, popsal jsem jeho klíčové vlastnosti, možné rozdělení a životní cyklus. Následně jsem se dotknul základních aspektů procesního řízení, zmínil jsem jeho definice, principy, přínosy a omezení procesního řízení. Kde to bylo možné, podpořil jsem slovní popis schematickým obrázkem pro ilustraci.

Kapitola 2
Modelování procesů

Druhá kapitola slouží k obeznámení se s modelováním procesů, které je nedílnou součástí životního cyklu procesů (viz podkapitola „**Životní cyklus procesů**"). Na začátek jsem zařadil úvod do modelování procesů za účelem seznámení s touto oblastí. Dále formuluji zásadní cíle a klíčové fáze modelování. Potom krátce popíšu principy modelování. Poslední části druhé kapitoly je seznámení se s pojmem standardu. Cílem kapitoly je postupně se obeznámit s oblastí modelování procesů, která je náležitě využitá v praktické části.

2.1 Úvod do modelování procesů

V úvodu do modelování procesů bych chtěl uvést formální definice pojmů modelování procesů a procesní model, který je výstupem modelování. Níže uvedené definice výstižně specifikují popisovanou oblast:

„*Modelování obchodních procesů je lidská činnost vytváření modelů obchodních procesů. Modelování obchodních procesů zahrnuje abstrakci od obchodního procesu z reálného světa, protože slouží určitému modelovacímu účelu. Proto pouze ty aspekty, které mají význam pro účely modelování, jsou zahrnuty do procesního modelu.*" [21]

„*Procesní model popisuje, typicky grafickou formou, aktivity, události a jejich propojení, které tvoří podnikový proces.*" [22]

Obecně je modelování procesů jedním z nástrojů, jak zlepšit kvalitu a efektivitu organizace. Základem modelování je popis procesu prostřednictvím různých prvků, které jsou součástí procesu (viz podkapitola „**Základní grafické prvky BPMN**"). Modelování procesů zpravidla popisuje logický vztah všech prvků procesu od jeho počátku až po dokončení v rámci jedné organizace. Ve složitějších situacích může modelování zahrnovat procesy nebo systémy mimo organizaci.

2.2 Cíle a fáze modelování

Modelování procesů má několik klíčových cílů: [23]

- **Popis procesů**: Prostřednictvím modelování lze sledovat, co se děje v procesech od začátku do konce. Modelování umožňuje získat interní a externí pohled na procesy a identifikovat vylepšení, která zvýší jejich účinnost a efektivitu.
- **Standardizace procesů**: Požadovaného výkonu a efektivity procesů lze dosáhnout díky řadě pravidel a pokynů pro provádění procesů, které stanoví modelování procesů.
- **Vytvoření propojení v procesech**: Modelování procesů jasně definuje vazbu mezi procesy a požadavky, které by tyto procesy měly splňovat.

Modelování procesů je druhou fází PDCA cyklu (viz podkapitola „**Životní cyklus procesu**“) a zpravidla zahrnuje implementaci několika po sobě jdoucích fází, mezi nimiž jsou: [24]

- Návrh AS-IS modelu – identifikace procesů a návrh původního modelu existujícího stavu organizace. Stanovení klíčových prvků a činnosti procesu. Výsledkem je vytvoření prvního draftu nebo původního modelu AS-IS.
- Revize, analýza a zlepšení AS-IS modelu – identifikace rozporu a duplicitních akcí v procesu, stanovení jeho omezení a vztahů. Výsledkem je konečná verze modelu AS-IS.
- Návrh modelu TO-BE – na základě předchozí analýzy určení modelu, který ukazuje, jak by měl proces vypadat v budoucnu včetně všech nezbytných vylepšení. Výsledkem je původní návrh modelu TO-BE, který se v této fázi vyvíjí.
- Testování a nasazení modelu TO-BE – nasazení modelu do produkčního prostředí, který je následně třeba otestovat. Na základě výsledků testování se může průběžně měnit.
- Optimalizace modelu TO-BE – v této fázi probíhá neustálé zlepšování procesů a tím pádem i procesního modelu.

2.3 Principy modelování

Modelování obchodních procesů je založeno na řadě klíčových principů, které umožňují vytvářet odpovídající procesní modely. Hlavní principy modelování jsou: [25]

- „*Rozděl a panuj*“ (lat. Divide et impera) – proces je reprezentován sadou hierarchicky uspořádaných prvků, v souladu s čímž by měl být proces rozdělen do jeho podstatných prvků.
- „*Úsporný zákon*“ (lat. Lex parsimoniae) – proces dekompozice a detalizace modelu by se měl řídit principem Occamové břitvy, který říká, že prvky se nemají zmnožovat více, než je nutné. [26]
- *Princip systematického přístupu* – považování činnosti za systém vzájemně propojených a vzájemně se ovlivňujících procesů.
- *Princip konzistence* – všechny prvky obsažené v procesním modelu by měly mít jednoznačnou interpretaci a neměly by si vzájemně odporovat.
- *Princip úplnosti* – před zahrnutím libovolného prvku do modelu je nutné vyhodnotit jeho vliv na proces. Pokud prvek není pro proces nezbytný, jeho zařazení do modelu se nedoporučuje.
- *Princip dokumentace* – prvky, které jsou součástí procesů, by měly existovat v modelu.
- *Princip ergonomie* – každý virtuální nebo skutečný list grafického popisu musí obsahovat takový počet objektů, aby bylo možné model snadno přečíst a analyzovat.

2.4 Modelovací standardy

Standardem se nazývá sada znaků a pravidel, která se používají ke grafickému popisu nebo modelování obchodních procesů, a jednoznačně definují interpretaci procesního modelu. Standard obecně určuje, jak je potřeba definovat procesy, aktivity, události atd., a podle jakých pravidel je spojujeme.

Na základě zadání bakalářské práce se v rámci příští kapitoly soustředím pouze na jeden standard: notaci BPMN (viz kapitola „**Modelovací notace BPMN**“).

2.5 Závěr druhé kapitoly

Ve druhé kapitole jsem se zaměřil na oblast modelování procesů, která je nezbytnou součástí logiky aplikace (viz kapitola „**Logika chování a uživatelské rozhrání aplikace**“). Kapitola byla sepsána především za účelem seznámení se s modelováním obchodních procesů, kterou jsem začal úvodem do oblasti, následně jsem zmínil důležité cíle modelování, po nichž jsem rovnou navázal na jednotlivé fáze modelování. Kromě toho jsem neopomněl uvést klíčové principy oblasti modelování procesu a závěrem jsem okrajově popsal standardy modelování.

Kapitola 3

Modelovací notace BPMN

Ve třetí kapitole se zaměřím na notaci BPMN, která je jedním z modelovacích standardů. Zpočátku se zabývám zásadní definicí a myšlenkami notace BPMN. Následně uvádím krátký náhled do její historie, který zároveň ilustruji obrázkem. Dále uvádím základní grafické prvky notace, které jsou nezbytné pro praktickou část. Dále se zajímám o strukturu notace BPMN a typy jejích diagramů. Záměrem této kapitoly je seznámení se s notací BPMN a její technickou strukturou.

3.1 Úvod do notace BPMN

Business Process Modeling Notation (dále jen „BPMN") je standard pro modelování procesů, jehož původní specifikace je vytvořena skupinou Object Management Group. Notace BPMN popisuje konvenci pro zobrazování procesů ve formě diagramů. Je to především metodika, která zmiňuje, jak správně použít elementy procesu v diagramu pro účel modelování, nikoliv jak správně popsat proces.BPMN notace obsahuje základní sadu intuitivních prvků, které umožňují definovat složité sémantické konstrukce. [6]

Klíčový cíl notace výstižně charakterizuje níže uvedená specifikace:

„*The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes.*" [6]

Z této specifikace lze odvodit, že důležitou vlastnosti BPMN notace je její zaměřenost na pracovníky z různých oblastí od technických specialistů, jako jsou vývojáři odpovědní za implementaci procesů, přes manažery, kteří kontrolují a řídí procesy, až po obchodní analytiky, kteří vytvářejí a zlepšují procesy.

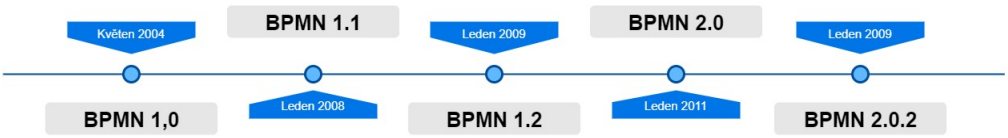
3.2 Náhled do historie BPMN

První verze BPMN 1.0 byla vydána v květnu 2004 společností Business Process Management Initiative (dále jen „BPMI“). Lze ji považovat za „zkušební verzi“, která měla omezené možnosti a vyžadovala četná vylepšení.

Další verze BPMN 1.1 byla vydána v lednu 2008. V tuto chvíli se vývojem a podporou zabývala organizace Object Management Group (dále jen „OMG“), která vznikla v důsledku sloučení s BPMI. Následující verze BPMN 1.2 se objevila o rok později v lednu 2009.

V lednu 2011 prezentovala společnost OMG verzi BPMN 2.0 a v prosinci 2013 byla vydána nejnovější verze BPMN 2.0.2. Dnes je BPMN jednou z nejběžnějších metod pro grafický popis podnikových procesů.

Klíčové etapy evoluce notace BPMN jsem znázornil na obrázku 3.1.



Obrázek 3.1: Náhled do historie BPMN

3.3 Základní grafické prvky BPMN

Procesy jsou v notaci BPMN modelovány pomocí sady grafických elementů. Podle [5] notace rozlišuje 4 hlavní skupiny objektů, které jsou využité během modelování:

- *Objekty toku (Flow Objects)* – události (events), aktivity (activities) a logické operátory (gateways).
- *Propojovací objekty (Connecting Objects)* – sekvenční tok (sequence flow), tok zpráv (message flow) a asociace (association).
- *Role nebo plavecké dráhy (Swimlanes)* – bazény (pools) a dráhy (lanes).
- *Artefakty (Artifacts)* – data (data), skupiny (groups) a textové anotace (text annotations).

V rámci této podkapitoly se zaměřím pouze na klíčové základní prvky, které jsou nezbytné k sestavení modelu v notaci BPMN.

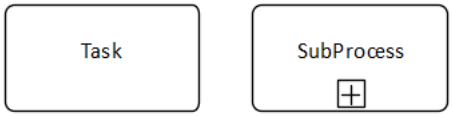
Objekty toku (Flow Objects):

- *Událost (Event)* popisuje konkrétní stav systému. Událost může mít spouštěč(trigger) nebo popsat určitý výsledek. K dispozici je kolem 30 typů událostí, které jsou rozdělené do třech klíčových skupin (viz obrázek 3.2):

- Zahajovací (Start)
- Průběžné (Intermediate)
- Konečné (End)

Obrázek 3.2: Klíčové události BPMN [5]

- *Aktivita (Activity)* je akce prováděná v procesu. Aktivita může být jednoduchá nebo složená (aktivita je vnořený podproces) (viz obrázek 3.3).



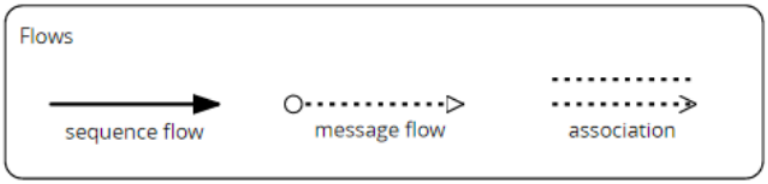
Obrázek 3.3: Activity BPMN [5]

- *Logický operátor (Gateway)* představuje body sjednocení (konvergence) nebo rozvětvení (divergence) pracovního toku (viz obrázek 3.4). Gateway lze považovat za bránu, která bud' pouští dál nebo nikoliv.

Obrázek 3.4: Logické operátory BPMN [5]

Spojovací objekty (Connecting objects):

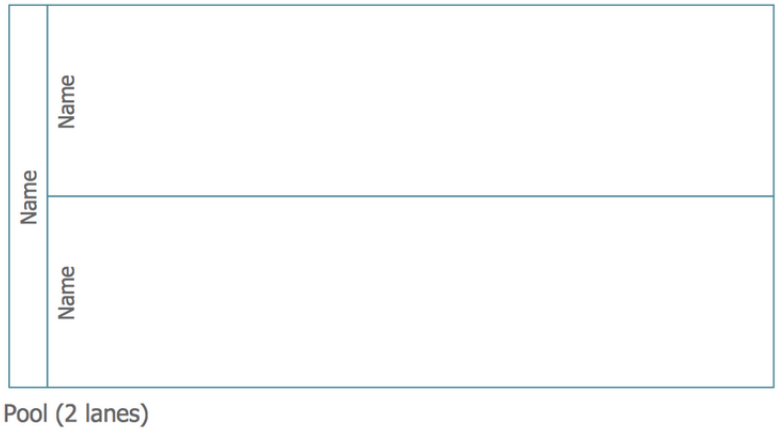
- *Úplná šipka* je tok aktivit, který ukazuje přenos řízení z aktivity na aktivitu a může procházet přes logický operátor (viz obrázek 3.5).
- *Tečkovaná šipka* je tok zpráv, který zobrazuje přenos dat mezi aktivitami a je samostatným tokem (viz obrázek 3.5).
- *Tečkovaná čára* je asociace, což jsou spojnice mezi anotacemi (dodatečné informace pro uživatele diagramu) a samotnými objekty (viz obrázek 3.5).



Obrázek 3.5: Spojovací objekty BPMN [5]

Role (Swimlanes):

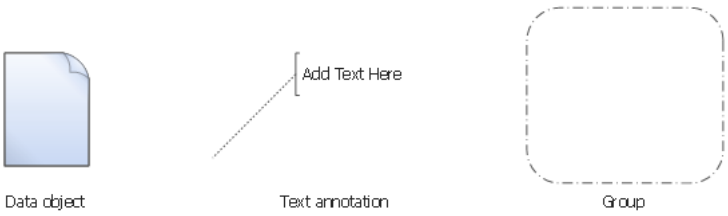
- *Swimlanes* organizují aktivity do skupin. Pomocí smimlanes lze rozdělit diagram na jednotlivé *dráhy (lanes)* nebo *bazény (swimlanes)* (viz obrázek 3.6). Každý bazén obsahuje aktivity, které provádí určitý subjekt. Dráhy člení bazén do více horizontálních částí.



Obrázek 3.6: BPMN bazén, který obsahuje 2 dráhy [5]

Artefakty:

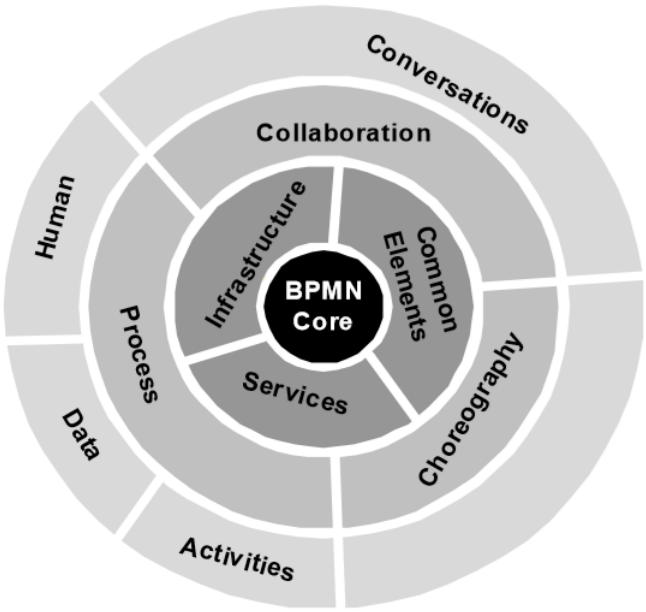
- *Artefakty* jsou objekty obsahující data, která jsou přenášena mezi aktivitami. Artefaktem je rovněž jakákoliv doplňující informace o objektech (viz obrázek 3.7).



Obrázek 3.7: BPMN artefakty [5]

3.4 Struktura BPMN a typy diagramů

Model struktury notace BPMN je založen na principu hierarchie vrstev, kde se každá vrstva staví na nižší vrstvu a rozšiřuje ji (viz obrázek 3.8). Součástí modelu je vnitřní jádro, které musí být jednoduché, stručné a rozšiřitelné. Jádro obsahuje nezbytné základní prvky, s jejichž pomocí lze diagram vytvářet [6].



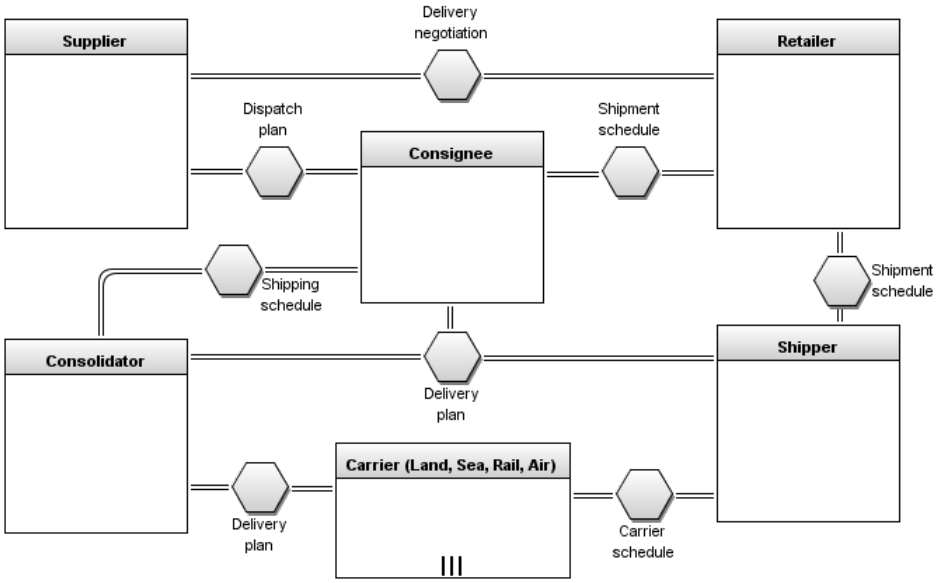
Obrázek 3.8: Model technické struktury BPMN [6]

Podle specifikace OMG [6] se dá diagram v notaci BPMN rozdělit na následující typy:

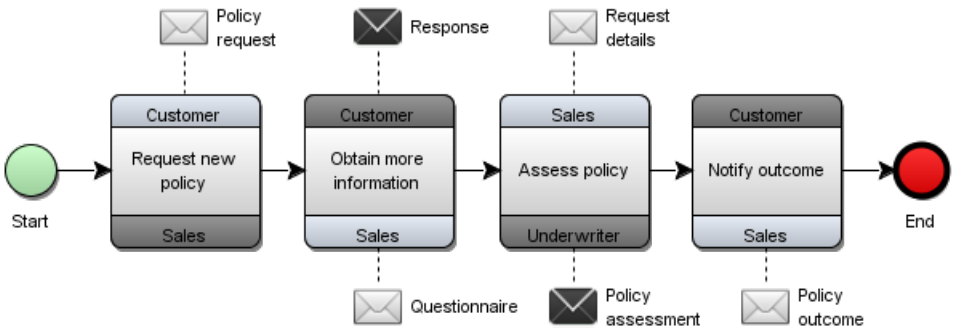
- Diagram **spolupráce** (collaboration), který se dělí na:
  - Diagram **konverzace** (conversation)
  - Diagram **choreografie** (choreography)
- Diagram **procesů**

Diagram **konverzace** zobrazuje interakce mezi účastníky formou toku zpráv a větvení. Představuje logiku procesu a nezabývá se jednotlivými úlohami procesu. Na obrázku 3.9 je znázorněn diagram, na kterém je představená konverzace v oblasti dodavatelského řetězce.

Diagram **choreografie** zobrazuje komunikace mezi účastníky a reprezentuje množinu výměn zpráv. Zaměřuje se na abstraktní pohled na proces, nikoliv na logiku procesu. Příklad využití diagramu choreografie je znázorněn na obrázku 3.10, kde je zobrazen proces žádostí o pojistku.



Obrázek 3.9: Příklad diagramu konverzace [7]



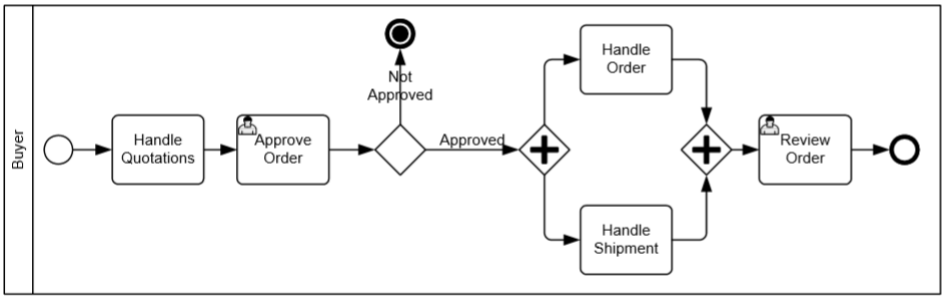
Obrázek 3.10: Příklad diagramu choreografie [7]

Diagram **procesů** popisuje sled nebo tok činností organizace, které splňují cíl vykonávání práce. Podle specifikace OMG je:

„*In BPMN a Process is depicted as a graph of Flow Elements, which are a set of Activities, Events, Gateways, and Sequence Flows that define finite execution semantics.*“ [6]

Příklad diagramu procesu je uveden na obrázku 3.11, na němž je ilustrován nákup z perspektivy kupujícího.

Na základě zadání bakalářské práce se v příštích kapitolách zaměřím pouze na diagram procesů.



Obrázek 3.11: Příklad diagramu procesů [6]

3.5 Závěr třetí kapitoly

Ve třetí kapitole jsem se věnoval notaci BPMN. Na začátku jsem krátce popsal, co je BPMN a jakou má specifikaci v oficiálních zdrojích. Následně jsem se krátce dotkl historie notace, kterou jsem podpořil obrázky. Potom jsem se zabýval popisem a ilustrací grafických prvků BPMN, které jsou nezbytné pro modelování libovolného procesu. Na závěr jsem zmínil vnitřní strukturu BPMN notace a charakterizoval možné typy BPMN diagramu.

Kapitola 4
Procesní analytik

V této kapitole se zabývám profesí procesního analytika. Na úvod posuzuji, jakým způsobem činnost procesního analytika navazuje na předchozí kapitoly a jak spolu souvisí. Dále představuji znalosti, jež jsou spojené s touto profesí. Následně formulují schopnosti, které jsou nezbytné k plnění úkolů profesního analytika. Cílem této kapitoly je uchopení praktického využití oblastí popsaných v předchozích kapitolách a plynulý přesun k další kapitole, která se už blíže týká praktické části této práce.

4.1 Profese procesní analytik

Povinnosti procesního analytika se velmi různí na základě specifik konkrétní společnosti. Tato profese úzce souvisí s oblastmi, jež jsem zmiňoval a probíral v minulých kapitolách.

Mezi klíčové činnosti procesního analytika patří analýza, určení a optimalizace podnikových procesů organizace (viz kapitola „**Procesní řízení a proces**“). Kromě toho je důležité mít jednoznačnou vizi a představu o činnostech firmy, což zpravidla znamená definovat AS-IS a TO-BE stav organizace a následně vybrat řešení, které by to umožňovalo. Tyto body jsou popsané v kapitole „**Modelování procesů**“.

4.2 Činnosti a požadavky procesního analytika

Při rešerši jsem se setkal s celou řadou různých činností procesního analytika, mezi nimiž jsou klíčové následující: [27][28]

- Sběr a analýza požadavků zákazníka
- Definování klíčových požadavků na produkt
- Popis a modelování podnikových procesů
- Analýza výkonu a následný návrh optimalizace procesů
- Prezentace výsledků práce

- Řízení požadavků – práce na jejich změnách, komunikace s vývojáři a zákazníky

Poté, co jsem prozkoumal různé nabídky práce, inzeráty a dostupnou literaturu ohledně požadavků na procesního analytika, došel jsem k následujícím závěrům. Nejčastěji se vyskytoval požadavek rozumět pojmu proces a jeho vlastnostem (viz podkapitola „**Proces**“), a mít znalosti v oblasti procesního řízení (viz podkapitola „**Základní aspekty procesního řízení**“). Velmi se cení praxe v oblasti sběru a popisu požadavků, rovněž určení a modelování podnikových procesů. Kromě toho velkou výhodou jsou zkušenosti z praktické aplikace nástrojů pro modelování. Velkou roli rovněž mají měkké dovednosti, beze kterých v profesi procesního analytika nelze efektivně používat odborné schopnosti a znalosti. Tyto dovednosti jsem stručně popsal v další kapitole.

4.3 Měkké a tvrdé dovednosti procesního analytika

Za účelem pochopení této podkapitoly, je důležité rozlišovat měkké dovednosti neboli soft skills, a tvrdé dovednosti neboli hard skills. Z toho důvodů jsem uvedl následující definici, kterou podle mého názoru výstižně charakterizují pojem soft skills:

„*Termín měkké dovednosti znamená, jak dobře se dokážete vypořádat nejen s lidmi a jejich způsoby chování, ale také se sebou samými. Dříve byly měkké dovednosti nazývány sociální kompetence. Tímto pojmem se ještě dnes označují faktory, jako jsou spolupráce, komunikace, schopnost zvládat konflikty*“.[29]

Pojem hard skills je výrazně snazší pochopit, protože představují odborné znalosti, dovednosti, zkušenosti, a tudíž i specifické znalosti odborného charakteru, které se dají určit podle stanovených měřítek.

Mezi *soft skills*, které jsou nezbytné pro procesního analytika patří: [30][31]

- Komunikační dovednosti – poněvadž většina informací se získává během živé komunikace, schopnosti aktivně naslouchat partnerovi, klást správné otázky a udržovat přátelskou atmosféru jsou velmi důležité.
- Schopnost řešit problémy – každý projekt je zaměřen na řešení konkrétních problémů. Úkolem analytika je definovat tyto problémy a určit množství práce na jejich odstranění.
- Kritické myšlení – schopnost vyhodnocovat informace, identifikovat problémy a formulovat úkoly.
- Schopnost rychle se učit – v praxi je téměř vždycky potřeba rychle se přizpůsobit určitému prostředí.
- Prostorová inteligence – procesní analytik by měl mít dobrou prostorovou představivost a měl by být schopen převádět informace do grafické

podoby.

Mezi *hard skills*, které jsou nezbytné pro procesního analytika patří: [32]

- Znalost systematického přístupu k analýze organizace
- Znalost různých metodik popisu podnikových procesů (ARIS, IDEF, BPMN)
- Znalost nástrojů pro popis a analýzu procesů
- Praktické zkušenosti s popisem podnikových procesů
- Zkušenosti s psaním předpisů, ustanovení a provozních pokynů
- Praktické zkušenosti s analýzou a zlepšováním procesů
- Praktické zkušenosti s určováním procesních indikátorů a metrik
- Znalost existujících referenčních modelů podnikových procesů
- Znalost klíčových procesů libovolné organizace

V souladu s cílem této práce jsem se v praktické časti zaměřil hlavně na tvrdé dovednosti.

4.4 Závěr čtvrté kapitoly

Ve čtvrté kapitole jsem se zabýval procesní analýzou. Začal jsem úvodem do této profese, ve kterém jsem zmínil, jak tato kapitola souvisí s procesním řízením a modelováním procesů. Následně jsem formuloval klíčové činnosti a popsal nutné požadavky pro profesi procesního analytika. Závěrem jsem specifikoval měkké a tvrdé dovednosti, které jsou nezbytné k plnění činnosti procesního analytika. V následující kapitole se budu věnovat možným způsobům zlepšení těchto dovedností a existujícím způsobům vzdělávání procesního analytika.

Kapitola 5

Způsoby vzdělávání procesního analytika

V páté kapitole se zabývám způsoby vzdělávání procesního analytika. Na začátku posuzuji, jakým způsobem lze zvýšit úroveň dovedností, které jsou nezbytné k plnění činnosti procesního analytika. Uvádím již existující možnosti a krátce je vzájemně porovnávám na základě výsledků, které získal při rešerši pro svou diplomovou práci *Jan Polan* [19]. Zaměřuji se na popis výhod a nevýhod různých přístupů, nikoliv na jejich konkrétní ukázky. Dále se zabývám popisem vlastních zkušeností, které jsem získal během absolvování univerzitního kurzu „Procesní řízení“ a obecně popisuji tento zdroj vzdělávání procesního analytika. Na závěr uvádím dva klíčové způsoby vzdělávání, jejichž pochopení je podstatné k postupnímu posunu směrem k praktické části.

5.1 Existující možnosti

Ve shodě s [19] v následném seznamu jmenuji jednotlivé klíčové způsoby vzdělávání, jejich krátký popis, a souhrn výhod a nevýhod:

- **Literatura**
  Literární zdroje se mohou lišit v jejich zaměření, mohou poskytovat různé metodologie a definice určitých pojmů, a obsahovat rozmanité příklady, což je podle mého osobního názoru spíš velkou výhodou díky možnosti se podívat na věci z různých úhlů. Navíc na většinu z těchto knih existují recenze, které popisují jejich důvěryhodnost. Rovněž se dá říct, že literatura obsahuje výstižný teoretický popis základních i pokročilých informací. Nicméně, nevýhodou takového zdroje vzdělávání je především časová náročnost při jejich čtení a vstřebávání.

- **Online kurzy**
  Cenově dostupných online kurzů je poměrně málo. Zpravidla takové kurzy mají velmi úzký rozsah a nízkou kvalitu. Na druhé straně sice existují výborně zpracované a hodnotné kurzy, ale ty jsou většinou velmi drahé. K očividným výhodám takového vzdělávání patří možnost studovat se z domova podle vlastních potřeb a tempa.

- **Komerční prezenční kurzy**
  Komerční prezenční kurzy jsou jedním z rychlých a intenzivních zdrojů

informací, které obvykle zaberou 2-3 dny. K výhodám takových kurzů lze přičíst osobní přístup ke každému studentovi, jazykovou přizpůsobivost, rychlé seznámení s oblastí a často i časovou flexibilitu. Navíc na konci student získává certifikát o úspěšném absolvování. Nicméně zpravidla takové kurzy mají velmi vysokou cenu. Kromě toho je obtížné najít opravdu kvalitní kurz a jeho intenzita může být nevýhodou v případě, že student neumí pracovat s velkým množstvím nových informací.

- **Hry**
  Jsou nejinteraktivnějším a pro většinu lidí možná nejzajímavějším způsobem vzdělávání. Výhodou tohoto způsobu je simulace reálných činností procesního analytika uživatelem. Takovým způsobem má uživatel možnost vyzkoušet si své teoretické znalosti při vykonávání určitých úkolů ve hře. Navíc v takovém prostředí uživatel dostává okamžitou zpětnou vazbu a časová náročnost je minimální. Účelem her je především seznámit uživatele s konkrétní oblastí procesního řízení. Z toho důvodu takové zdroje nemohou poskytnout široký teoretický základ.

- **Praxe**
  Toto je nejužitečnější a nejlepší způsob vzdělávání. Zájemce o procesní řízení získává pomocí praxe reálné zkušenosti, pracuje v reálném prostředí a, na rozdíl od her, s reálnými lidmi. Takovým způsobem lze samostatně pochopit výhody a nevýhody procesního řízení. Kromě toho bych řekl, že je to jediný způsob, který zlepšuje a zapojuje oboje, jak měkké dovednosti, tak i tvrdé. Nevýhodami tohoto způsobu jsou náročnost a stres, který pro někoho může být ze začátku nepříjemným a zastrašujícím faktorem.

- **Univerzitní prezenční kurzy**
  Jedná se o podobný přístup, jako při výuce komerčního kurzu s několika podstatnými rozdíly. Je podrobněji popsán v následující podkapitole „Univerzitní kurz“.

5.2 Univerzitní kurz

Tento zdroj vzdělávání patří, stejně jako komerční kurz, k prezenčním zdrojům. Je tu ale několik podstatných rozdílů, které popíšu v dalším odstavci. Předtím než se do toho ale pustím, chtěl bych zdůraznit, že univerzitní kurzy se mohou výrazně lišit. Následující popis se opírá o zkušenosti, které jsem získal při studiu předmětu „Procesní řízení“ na ČVUT. Stejný kurz absolvoval i autor *Jan Polan*, který provedl hlubší průzkum existujících možností vzdělávání [19].

Prvním a podle mého osobního názoru nejdůležitějším rozdílem jsou kvalita poskytovaných informací a odbornost vyučujících, které jsou zpravidla mnohem lepší než v komerčních kurzech. Dále univerzitní kurzy mohou přitahovat k výuce i odborné externisty, kteří se se studenty mohou podělit o své zkušenosti z praxi, což je samozřejmě velkým přínosem pro každého účastníka kurzu.

Druhým rozdílem je doba trvání, která je v tomto případě mnohem delší. Nové informace jsou podávány postupně, což umožňuje studentům lépe se v nich zorientovat a přemýšlet o nich. Díky tomu se zlepšuje zpětná vazba, která může být poskytnutá vyučujícími. Na druhou stranu pro lidi, kteří se potřebují rychle seznámit s oblastí procesního řízení, může být taková postupná výuka nevhodná pro svou zdlouhavost i přesto, že je její kvalita na vysoké úrovni.

Třetí rozdíl je v nutnosti splnit požadavky univerzitního kurzu. Nejčastěji je pro pokračování ve studiu nutné absolvovat předměty, které si student zapsal. To může být oboje výhodou i nevýhodou. Na jedné straně má díky tomu student motivaci splnit takový kurz a opravdu se musí naučit novým věcem. Na druhou stranu pro někoho to může být omezením, pokud studijní plán neodpovídá jeho osobním cílům či přáním.

Čtvrtý rozdíl je v zaměření na týmovou práci. Komerční kurzy zpravidla pojímají menší skupiny než univerzitní, což je způsobeno jejích intenzitou. Univerzitní kurzy navíc často vyžadují plnění týmových semestrálních projektů, které zabírají celé měsíce. Díky tomu má student možnost rozvinout své měkké dovednosti, které jsem zmiňoval v podkapitole „**Měkké a tvrdé dovednosti procesního analytika**".

5.3 "Just-in-case" a "just-in-time" učení

Obecně se dá způsob vzdělávání rozdělit na dva vymezující se přístupy: *just-in-case learning* neboli učení se „do zásoby" a *just-in-time learning* neboli učení se „za chodu".

Just-in-case learning je tradiční a obvyklý způsob učení, při kterém se učíme pro případ, že danou vědomost budeme moci využít později. Problematické je samozřejmě to, že člověk neví, zda dané vědomosti vůbec bude někdy potřebovat. Je také možné, že ve chvíli, kdy bude tyto informace potřebovat, budou již zastaralé. Tento typ učení funguje dobře ve školním prostředí, kde se naučené informace ověřují prostřednictvím testování. Většina takových informací je brzy zapomenutá, pokud si je člověk pravidelně neoživuje. Největší nevýhodou takového přístupu je nejčastěji snaha studenta materiál si především zapamatovat, aniž by se snažil pochopit jeho základy a podstatu.[33][34]

Just-in-time učení je opakem učení se do zásoby. Při tomto pojetí vzdělávání se studenti mohou dostat k informacím které potřebují právě ve chvíli, kdy je potřebují. Takovým způsobem se rovnou uplatní nové znalosti v praxi, tím pádem si člověk tyto znalosti lepe zapamatuje. Pro výstižnější specifikaci uvádím následující úryvek:

„*V procesu vzdělávání je efektivnější (z hlediska motivace, paměti, zájmu...) studovat to, co je pro daný problém potřebné (to je východisko pragmatické pedagogiky) a věnovat se reálným problémům, které řeší daný studující. Znalosti, dovednosti a informace jsou získávány jako síť, nikoli jako uspořádaný řetězec. Klade se důraz na učení ze zkušenosti, a to jak individuální, tak také sociální.*" [35]

Z důvodů, že mým hlavním cílem je poskytnout aplikaci, která bude doplněním k existujícím možnostem vzdělávání a která by se zároveň skutečně blížila k simulování reálného prostředí procesního analytika a umožňovala by uživateli vyzkoušet si své teoretické znalosti v praxi, jsem se zaměřil právě na druhý způsob vzdělávání, tedy just-in-time learning. Podrobněji se požadavkům na takovou aplikaci věnuji v příští kapitole „**Analýza a návrh požadavků**“.

5.4 Závěr páté kapitoly

V páté kapitole jsem se zaměřil na rozličné způsoby vzdělávání procesního analytika. Na začátku jsem se věnoval již existujícím možnostem vzdělávání procesního analytika, zhodnotil jsem jejich výhody a nevýhody. Následně jsem se zastavil nad možnostmi vzdělávání na univerzitním kurzu, kde jsem se podrobně zaměřil na výhody a nevýhody takového přístupu. Neopomněl jsem také zmínit své vlastní zkušenosti z absolvování takového univerzitního kurzu. Potom jsem se zabýval typy vzdělávání just-in-time a just-in-case, kde jsem stručně vyhodnotil jejich efektivitu. Na závěr jsem uvedl typ vzdělávání, který jsem zvolil ve své praktické části. V příští kapitole se již budu věnovat požadavkům na aplikaci, která by umožnila studentům efektivně podpořit a doplnit zdroje vzdělávání se v procesním řízení.

Kapitola 6
Analýza a návrh požadavků

V této kapitole se zabývám definováním obecných a technických požadavků na aplikaci, která by umožnila zájemcům o procesní řízení vyzkoušet si prakticky své znalosti ve světě modelování v notaci BPMN. Na základě výsledků prozkoumání již existujících způsobů vzdělávání procesního analytika a zadání bakalářské práce jsem dospěl k závěru, že by tato aplikace mohla sloužit jako doplněk k těmto řešením.

6.1 Účel aplikace

Hlavním účelem aplikace bylo vytvořit nové interaktivní prostředí (simulátor), které bude doplňkem k již existujícím zdrojům vzdělávání procesních analytiků. Student, který již má nějaké základní znalosti v oblasti procesního řízení a analýzy, má možnost si prakticky vyzkoušet typické úkoly, se kterými se procesní analytik běžně setkává. Taková aplikace umožní člověku snížit rozdíl mezi jeho teoretickými znalostmi a praktickými dovednostmi.

Můj návrh je založen na alternativním způsobu vzdělávání „just-in-time“, což umožní studentovi intuitivně si vyzkoušet procesní analýzu a následné modelování a zároveň se naučit novým teoretickým věcem. Tento způsob povede k lepšímu uplatnění teoretických znalostí a k rychlejšímu získání dovedností, které jsou nutné pro práci procesního analytika.

6.2 Obecné požadavky

Uživatel formou simulace ověří svou schopnost identifikovat ve slovním popisu správné elementy notace BPMN a propojit je do modelu, který dodržuje specifikaci notace a reflektuje slovní popis. Aplikace zkontroluje, jestli uživatel identifikoval elementy notace BPMN a následně převedl slovní popis procesu do podoby modelu v notaci BPMN správně. Pokud vytvořený model nebude odpovídat slovnímu popisu nebo specifikaci notace BPMN, bude uživatel upozorněn na chyby a na chybějící části. V případě, že uživatel zůstane stát na jednom bodě, bude mít možnost využít nápovědu. Absolvování všech úkolů simulátoru by mělo trvat od 20 do 45 minut, v závislosti na dovednostech a teoretických znalostech uživatele.

6.2.1 Cílová skupina

Cílovou skupinou uživatelů jsou lidé libovolného věku, kteří se zajímají o procesní řízení a analýzu, mají v této oblasti alespoň nějaké znalosti a chtějí si vyzkoušet činnosti s ní spojené. Během průchodu aplikací si uživatel postupně uvědomuje, jaké dovednosti by měl mít procesní analytik a na konci získává zkušenosti z procesní analýzy a modelování procesů v notaci BPMN.

Takový návrh simulátoru je vhodný jak pro začátečníky, kteří budou mít možnost učit se v praxi formou simulace, tak pro pokročilejší uživatele, kteří již mají jisté zkušenosti v oblasti procesní analýzy a modelování. Pro pokročilejší uživatele může aplikace sloužit jako nástroj na ověření a oživení již získaných znalostí nebo jako rozcvička při přípravě na náročnější výkon.

6.2.2 Části aplikace a use casy

Uživatel dostává informace o procesech formou předem připraveného slovního popisu. V závislosti na části aplikace má uživatel buď předem definovanou nápovědu ve formě tučného písma, která ho upozorňuje na konkrétní data, nebo musí tato data identifikovat sám. Po přečtení a analýze slovního popisu musí uživatel poskytnout zpětnou vazbu formou vstupních dat, jejichž účel se liší v závislosti na fázi. Simulátor následně ověří správnost těchto vstupních dat a buď uživatele pustí do další fáze aplikace, nebo poukáže na případné chyby.

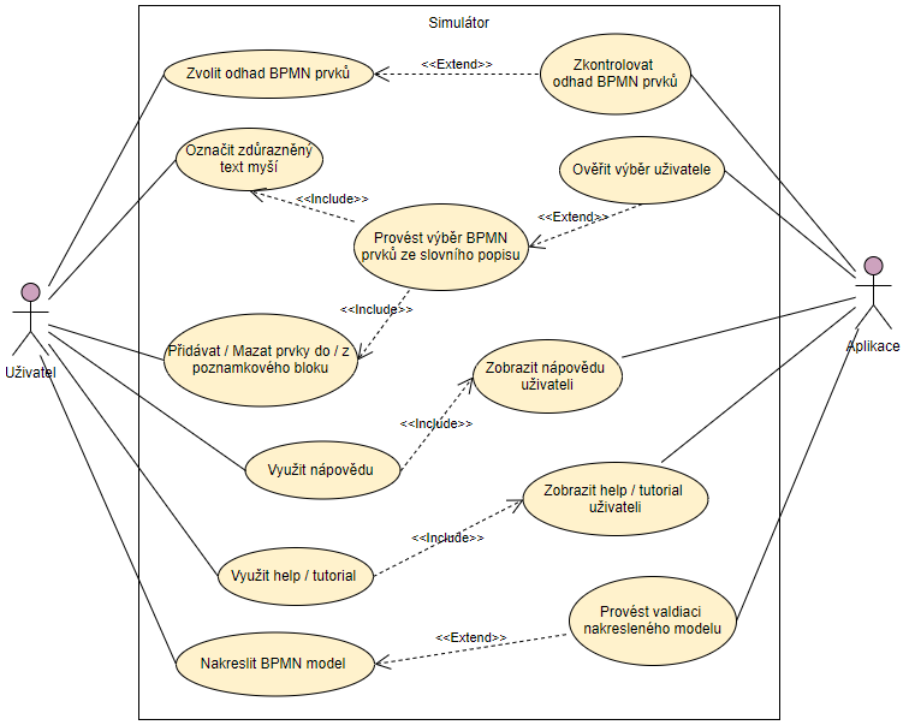
Po ověření správnosti identifikace elementů nutných pro modelování BPMN může uživatel začít modelovat procesy zakreslením v okně s kreslicím nástrojem BPMN. Cílové procesy jsou sestavené na základě prvotního popisu procesů v úvodní fázi simulátoru.

Základní případy použití jsou zobrazené na obrázku 6.1.

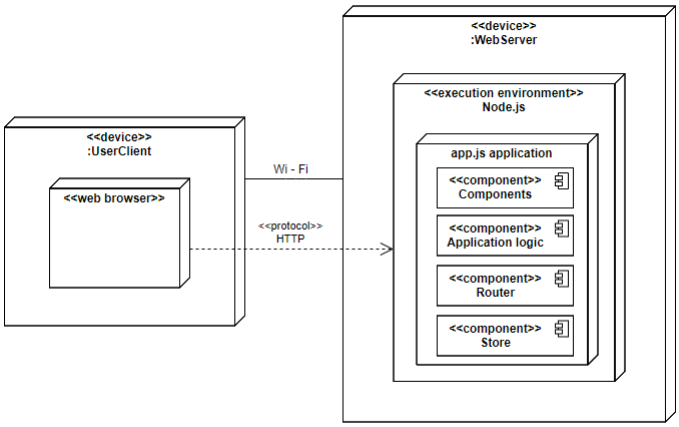
6.3 Technické požadavky

6.3.1 Obecné požadavky

Hlavním úkolem bylo vytvořit takovou aplikaci, která bude primárně odpovídat současným technologiím, bude univerzální, snadno přístupná z webu, dostupná a rozšiřitelná. Taková aplikace by měla mít snadné nastavení a její užití by mělo být každému uživateli jasné a srozumitelné hned při prvním spuštění. Jednoduchost nasazení takové aplikace ukazuje diagram nasazení na obrázku 6.2.



Obrázek 6.1: Diagram případů použití



Obrázek 6.2: Diagram nasazení

6.3.2 Programovací technologie

Na základě obecných požadavků je vhodným řešením webová aplikace, která nebude vyžadovat použití externích backend technologií ani databází, a přístupná z webového prohlížeče.

Pro zjednodušení vývoje jsem se rozhodl použít JavaScript framework. Podle výzkumů [36] jsem došel k závěru, že v dnešní době jsou nejpopulár-

nější tři JavaScript frameworky. Pro porovnání jsem je srovnal do tabulky 6.1, která mi pomohla při výběru vhodného frameworku.

	Angular.js	**React.js**	**Vue.js**
Úroveň složitosti	Vysoká	Střední	Nízká
Podrobnost dokumentace	Podrobná	Průměrná	Velmi podrobná
Velikost souboru	500 kb	100 kb	20 kb
Doba načtení	Dlouhá	Střední	Krátká
Výkonnost	Průměrná	Vysoká	Vysoká
Nejvíc vhodný na	Velké projekty	Střední a malé projekty	Střední a malé projekty
Zpětná kompatibilita	Špatná – závislost na minulé verzi	Úplná kompatibilita a výborná aktualizace	Úplná kompatibilita a výborná aktualizace
Zahájen v roce	2010	2013	2014
Využití	Google, Wix	Facebook, Uber	Alibaba, Gitlab

Tabulka 6.1: Srovnání JavaScript frameworku

Zvolil jsem framework **Vue.js**, který jsem preferoval před frameworky React.js a Angular z několika hlavních důvodů. Především je to jednoduchost ovládání. Framework Vue.js má jednoduché základy a velmi podrobnou dokumentaci, což může výrazně urychlit proces učení a ušetřit tak čas na vývoj aplikace.

Důležitou vlastností je vysoká škálovatelnost Vue.js, která umožňuje jednoduše provádět změny v aplikaci. Navíc má framework výbornou zpětnou kompatibilitu, která je velmi důležitá pro budoucí aktualizaci a upgrade aplikace. Poslední prvek, který mě v mém rozhodnutí přesvědčil, byla malá velikost frameworku, která tak umožňuje udržovat vysoký výkon aplikace a zároveň zachovat její vysokou funkčnost. Framework Vue.js poskytuje možnost přizpůsobení, které umožňuje kombinovat chování uživatelského rozhraní a komponent v jednom skriptu. Kromě toho umožňuje integraci různých knihoven, jako je Bootstrap, což značně usnadňuje proces vývoje. Na základě těchto vlastnosti jsem si pro svou aplikaci zvolil framework Vue.js.

Kromě samotného frameworku jsou v aplikaci využité následující technologie:

HTML

HTML je značkovací jazyk, s jehož pomocí se vytváří struktura neboli kostra webových stránek prostřednictvím různých prvků. Používá se k tvorbě těla téměř všech webových stránek a prohlížeče jej snadno interpretují. Poslední vydaná verze je HTML 5.

CSS

CSS (Cascading Style Sheets) je jazyk, který popisuje grafické zobrazení

jednotlivých HTML prvků. Zatímco HTML tvoří strukturu webové stránky, CSS definuje její vzhled. Jedná se o jednu ze základních technologií moderního internetu. Téměř žádný web se neobejde bez CSS, což znamená, že HTML a CSS fungují společně. Nejaktuálnější verze CSS je 3.

BootstrapVue
Bootstrap je populární HTML / CSS knihovna pro vytváření adaptivních webových stránek, které podporuje většina moderních prohlížečů. Pro své základní funkce však obvykle používá knihovnu jQuery, stejně jakož i rozsáhlý seznam komponent, jako jsou upozornění a modální prvky. Vzhledem k tomu jsem se rozhodl pro technologii BoostrapVue, která podporuje nejen komponenty Bootstrap a gridový systém, ale zahrnuje také podporu směrnic Vue.js, což poskytuje celou řadu funkcí z ekosystému frameworku Vue.js. BootstrapVue umožňuje vytvářet adaptivní, mobilní a ARIA přístupné projekty pomocí Vue.js a nejoblíbenější knihovny rozhraní na světě – Bootstrap.

Tato kapitola zahrnovala požadavky na aplikaci, která by se měla stát vhodným a efektivním doplněním ve vzdělávání procesních analytiků. V další kapitole podrobně popíšu logiku chování, možnosti a funkcionalitu aplikace.

Kapitola 7

Logika chování a uživatelské rozhraní aplikace

V této kapitole se zabývám celkovou strukturou aplikace, její logikou a funkcionalitou. Zaměřím se na popis fungování, uživatelské rozhraní a validaci vstupních dat. Nejdřív popíšu strukturu aplikace a její rozdělení na bloky. Dále se zaměřím na popis funkčnosti a logiky každého bloku, a v rámci této logiky uvedu několik ukázek uživatelského rozhraní, ve kterém budou probíhat činnosti. Nakonec vysvětlím, jakým způsobem probíhá validace vstupních dat v různých blocích aplikace, a co je potřeba udělat pro její úspěšné dokončení.

7.1 Struktura aplikace

Koncept této aplikace spočívá především ve vytvoření interaktivního prostředí – simulátoru, ve kterém bude uživatel schopen aplikovat své znalosti v oblasti procesní analýzy a modelování v praxi. Aplikace umožní ověřit takové dovednosti, jako jsou prozkoumání a analýza slovního popisu, schopnost zdůraznit a vyčlenit důležité věci z textu, schopnost rozlišovat BPMN elementy podle jejich typu, identifikovat jednotlivé procesy a v nich obsažené BPMN elementy, a nakonec dovednost tyto procesy modelovat.

7.1.1 Rozdělení na bloky

Aby měl uživatel možnost seznámit se s každou z výše uvedených dovedností, je funkčnost simulátoru rozdělená do třech bloků, které se nazývají fáze. V nichž má uživatel příležitost soustředit se na konkrétní oblasti procesní analýzy nebo na modelování procesů a vyzkoušet své dovednosti v praxi. Jednotlivým fázím se budu věnovat v následujících podkapitolách (viz podkapitoly **7.2**, **7.3** a **7.4**).

7.1.2 Koncepce uživatelského rozhraní

Vzhledem k tomu, že aplikace je rozdělená do třech bloků, kde účelem každého bloku jsou především simulace úkolu z činnosti procesního analytika (viz kapitola „**Procesní analytik**“), možnost vyzkoušení a ověření teoretických

znalostí uživatele z procesní analýzy v praxi, je uživatelské rozhraní navržené tak, aby bylo co nejjednodušší. UI je udělané tak, aby se v něm uživatel rychle zorientoval a dokázal se soustředit na konkrétní zadání. Veškeré ovládání aplikace dělá uživatel pomocí myši.

7.1.3 Validace vstupních dat obecně

Na základě toho, že hlavním účelem aplikace je poskytnout uživateli kvalitní zdroj vzdělávání procesních analytiků, podstatnou věcí je správné logické zpracování vstupních dat uživatele. Aplikace musí být schopna ověřit, zda jsou určité úkoly správně splněny. Tato kontrola se liší v závislosti na fázi a typu vstupních dat. Kontrola se skládá z několika na sebe navazujících kroků. V případě správné validace všech bodů lze fázi počítat za splněnou a uživatel dostane gratulaci. V opačném případě se mu objeví chybové hlášení.

7.2 Fáze 1 – předběžná analýza

7.2.1 Funkčnost a uživatelské rozhraní

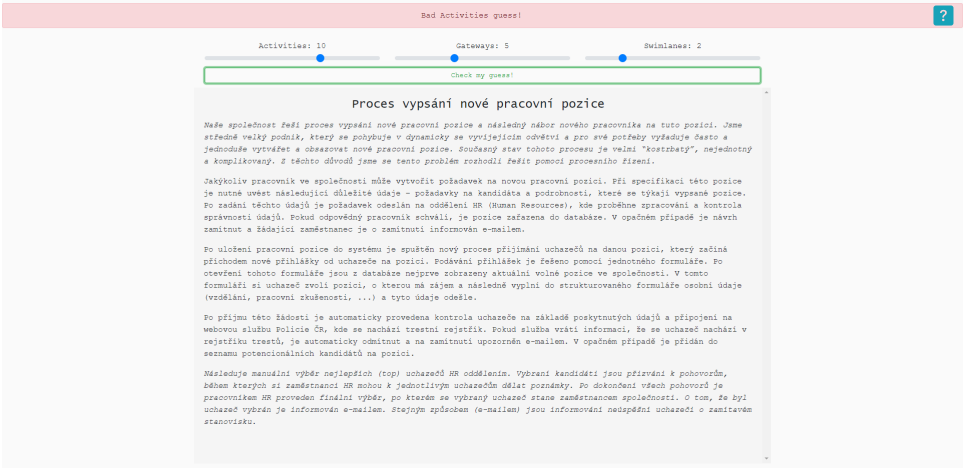
První blok aplikace je fáze předběžné analýzy. V ní se uživatel setká s předpřipraveným slovním popisem procesu konkrétní firmy, který se skládá z několika dalších podprocesů. Slovní popis neobsahuje žádné nápovědy a zdůraznění textu. Každý z podprocesů zahrnuje potenciální BPMN prvky – swimlany (swimlanes), události (events), aktivity (tasks) a rozhodovací logické operátory (gateways). Definice jednotlivých prvků je uvedena v kapitole „**Modelovací notace BPMN**“.

Úkolem uživatele je přečíst si tento slovní popis, provést jeho krátkou analýzu, najít procesy a zkusit odhadnout, kolik různých BPMN prvků text obsahuje. Jelikož si myslím, že je velice těžké rychle ze slovního popisu pochopit, kde jsou události a kde aktivity, rozhodl jsem se spojit je do jednoho typu – activities. To znamená, že uživatel musí definovat, kolik activities (tasks + events), swimlanes a gateways obsahuje slovní popis.

Po vyplnění odhadu do vstupní formy, aplikace ověří jeho správnost a vrátí uživateli výsledek, kterým ho buď informuje, že je vpuštěn do další fáze simulátoru, anebo ho upozorní na chybu pomocí chybového hlášení (viz obrázek 7.1). Uživatel má pět pokusů na odhadnutí počtu BPMN prvků. Pokud v tom neuspěje, dostane upozornění se správnou odpovědí a aplikace ho vpustí do další fáze. Jde o to, aby uživatel nebyl demotivován opakovaným neúspěchem. Hlášení a upozornění jsou podrobněji popsané v kapitole „**Vývoj**“.

7.2.2 Kontrola odhadu BPMN prvků

V první fázi předběžné analýzy je realizována velmi jednoduchá kontrola, během které aplikace ověří, jestli uživatelův odhad BPMN elementů, odpovídá předem definovanému správnému počtu těchto elementů.



Obrázek 7.1: Ověřování odhadu ve fázi předběžné analýzy

7.3 Fáze 2 – analýza slovního popisu

7.3.1 Funkčnost a uživatelské rozhraní

Potom následuje fáze analýzy slovního popisu a BPMN prvků. V předchozí fázi musel uživatel pouze odhadnout, kolik obsahuje text určitých elementů. V této fázi se zaměří na dovedností analyzovat slovní popis a rozlišovat BPMN prvky podle jejich typu.

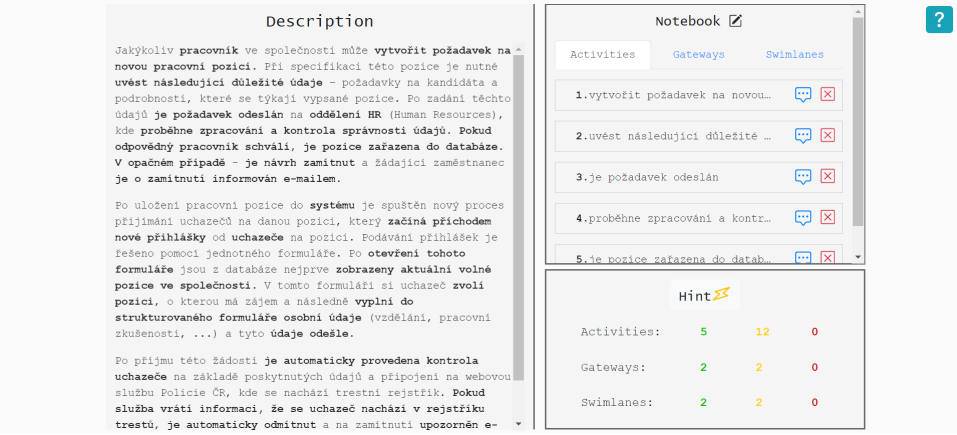
Plocha aplikace je rozdělená do třech komponent (částí): slovní popis, poznámky a nápověda. Každá z těchto částí je dynamicky spojená s ostatními a aktualizuje se podle celkového stavu aplikace. Komponenty jsou podrobněji popsané v kapitole „**Vývoj**“.

Na rozdíl od předchozí fáze předběžné analýzy jsou ve slovním popisu tučným písmem zdůrazněné fráze, na které by uživatel měl dávat pozor. Každá z těchto frází je jednotlivým BPMN elementem. Uživatel má za úkol rozlišit tyto elementy podle třech mnou definovaných typů: activities, gateways a swimlanes.

Funguje to tak, že uživatel označí jakoukoliv zdůrazněnou frázi pomocí myši, následně stiskne pravé tlačítko myší a hned se mu otevře context menu, ve kterém má na výběr právě ze tří možností: activities, gateways a swimlanes (viz obrázek 7.2). Jakmile zvolí jednou z těchto možností, objeví se mu označená fráze v poznámkách.

Poznámkový blok se nachází vpravo nahoře a obsahuje tři odvětví poznámek, které jsou rozdělené podle typu BPMN elementů. Mezi těmito větvemi lze přepínat a z poznámek lze odebírat BPMN prvky.

Pod poznámkovým blokem se nachází nápověda, která je uspořádána podle typu BPMN prvků. Každý typ má aktuální informace o tom, ve kterém stádiu se nachází uživatel. To znamená, že blok nápovědy obsahuje čtyři sloupce, kde první sloupec je název typu, druhý je počet správných zařazení do určitého typu elementů (na obrázku 7.2 zelenou barvou), třetí je počet chybějících

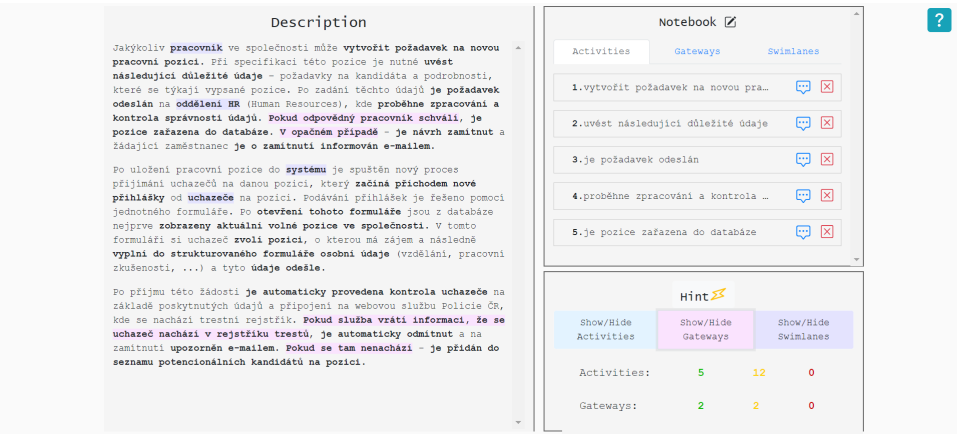


Obrázek 7.2: Označení a přidání BPMN prvků do poznámek podle jejich typu

prvků tohoto typu (na obrázku 7.2 žlutou barvou) a čtvrtý je počet špatně zvolených prvků (na obrázku 7.2 červenou barvou).

Nápověda se dynamicky obnovuje podle aktuálního stavu poznámek, což znamená, že jakmile uživatel zařadí určitý BPMN prvek ze slovního popisu do poznámek, nápověda se hned aktualizuje. Díky tomu může uživatel kdykoliv zkontrolovat, jestli je jím provedená analýza správná.

Kromě toho má uživatel možnost využít další nápovědu, která mu ve slovním popisu barevně označí, jak by měly být uspořádány BPMN elementy podle jejich typu (viz obrázek 7.3). Tuto nápovědu lze ale využít jen v případě, že uživatel má již alespoň 50 procent správně zařazených prvků.



Obrázek 7.3: Využití barevné nápovědy

Důležité je uvědomit si, že stejný prvek nelze přidat opakovaně do poznámek, ať už do stejného či odlišného typu. Pokud se o to uživatel pokusí, objeví se chybové hlášení (viz podkapitola „**Ověření správnosti zařazení BPMN elementů**“). Další prvek znemožňuje uživateli vybírat „Swimlanes”, pokud už jsou všechny swimlanes správně zařazené. Tato možnost zmizí z context menu.

Aplikace dynamicky ověřuje po každém zařazení, jestli už má uživatel

správně uspořádané poznámky. Jakmile se to stane, simulátor pogratuluje uživateli a nasměruje ho do další fáze.

7.3.2 Ověření správnosti zařazení BPMN elementů

Druhá fáze je složena z řady různých validací.

První je kontrola označování správné části slovního popisu. Simulátor zkontroluje, zda označený text odpovídá jednomu z těch, které jsou zdůrazněné tučným písmem. Pokud se uživatel snaží přidat nějaký jiný text nebo má označenou necelou frázi, dostane upozornění, že má označovat jenom text, jehož písmo je tučné.

Druhou je kontrola duplicitního zařazení. Aplikace ověří, jestli se prvek, který chce uživatel přidat, ještě nenachází jinde v poznámkovém bloku. V případě nálezu duplicity uživatel dostane upozornění, že tento element už existuje v poznámkách, a duplicitní zařazení tak nenastane.

Třetí je kontrola kompletnosti elementů podle typu. To znamená, že aplikace ověří, jestli počet správných prvků určitého typu odpovídá počtu nutných prvků tohoto typu. Pokud ano, z context menu, které se objeví po stisknutí pravého tlačítka myši, zmizí možnost zařadit element do tohoto typu.

Čtvrtou je validace přítomnosti a správnosti umístění nutných elementů, které jsou zvýrazněné tučným písmem ve slovním popisu, v poznámkovém bloku. Tato kontrola ověří, jestli prvek, který uživatel zařadil do poznámek, odpovídá svému reálnému typu. Pokud ano, do bloku nápovědy se přidá jeden prvek do počtu správných a počet chybějících prvků se sníží o jeden, v opačném případě se o jeden zvýší počet špatných a zůstane stejný počet chybějících prvků.

Poslední pátou kontrolou je ověření správného zařazení všech nutných prvků. Simulátor zkontroluje, zda počet správných prvků určitého typu odpovídá počtu nutných prvků tohoto typu a počet ostatních (chybějících a špatných) se rovná nule. Tato validace se dělá automaticky po jakékoliv aktualizaci poznámkového bloku. V případě úspěchu uživatel uvidí modální okno s gratulací a možností postupu do další fáze.

7.4 Fáze 3 – Modelování a s ním spojené činnosti

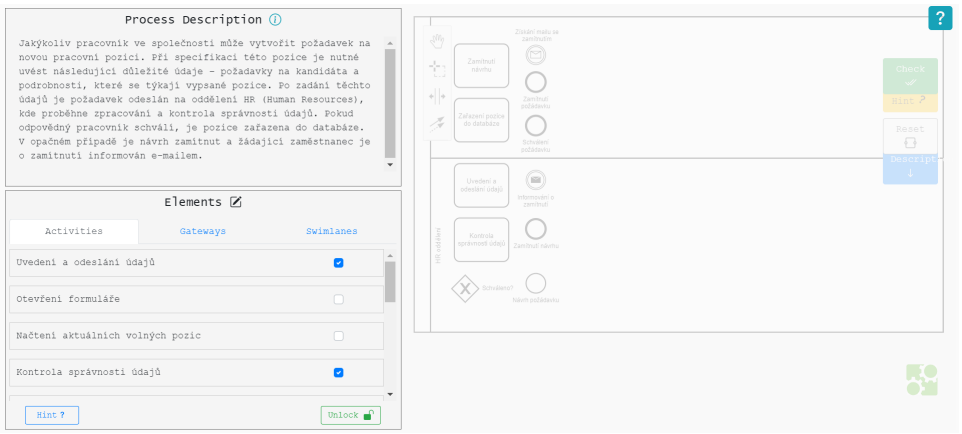
7.4.1 Funkčnost a uživatelské rozhraní

Třetí fáze je rozdělená do dvou částí.

První část – checklist

V první části uživatel uvidí na ploše obrazovky tři hlavní komponenty: popis konkrétního podprocesu ze slovního popisu druhé fáze, checklist(seznam) BPMN elementů a modelovací okno, které je ze začátku neaktivní. V této části je úkolem uživatele přečíst popis procesu a následně vybrat prvky z checklistu, které patří k tomuto procesu (viz obrázek 7.4). Z procesního hlediska tímto způsobem ověří simulátor schopnost uživatele porozumět popisu

jednotlivých procesů a identifikovat v nich obsažené BPMN elementy. V této části je důležité si uvědomit, že aplikace povolí použít nápovědu pomocí tlačítka „Hint“ jen v případě, že uživatel má již správně označených aspoň 40 procent prvků z checklistu. Pokud tomu tak je, správné prvky se označí zelenou barvou.



Obrázek 7.4: Identifikace elementů podle popisu procesu

Po stisknutí tlačítka „Unlock“ následuje jednoduchá kontrola checklistu (viz podkapitola „**Validace modelování a checklistu**“) a pokud je úspěšná, aplikace pogratuluje uživateli a odblokuje modelovací okno, které rozšíří na celou obrazovku. V tuto chvíli se modelovací okno aktivuje a uživatel může začít druhou část třetí fáze, tedy fázi modelování procesu.

Druhá část – modelování

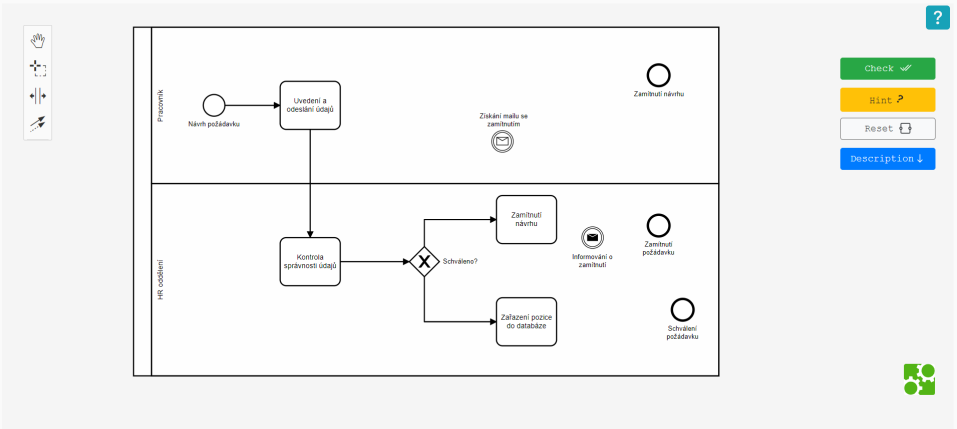
V druhé části fáze modelování se uživatel dostane ke kreslicímu nástroji bpmn.io. Tomuto nástroji se budu věnovat v kapitole „**Vývoj**“. Uživatel má předpřipravené BPMN elementy na pracovní ploše, které je nutné správně propojit a případně umístit do příslušných swimlanes. Úkolem uživatele je dosáhnout náležitého spojení všech BPMN prvků, které jsou již na diagramu, nikoliv přidávat nebo mazat jakékoliv elementy.

Kromě kreslicího nástroje jsou na ploše také čtyři tlačítka: Check, Hint, Reset a Description (viz obrázek 7.5). Pomocí tlačítka „Description“ může uživatel otevřít popis procesu, který je nutné nakreslit. „Reset“ vrací BPMN diagram do jeho původního stavu. „Hint“ umožňuje uživateli dostat nápovědu, jak by měl postupovat dál. Využít ji je možné pouze v případě, že uživatel dosáhl minimálně 40 procent správných spojení. Pomocí tlačítka „Check“ aplikace provede validaci celého procesního diagramu. Pokud budou zachyceny chyby simulátor upozorní uživatele chybovým hlášením. V opačném případě pogratuluje uživateli a simulace se ukončí.

7.4.2 Validace modelování a checklistu

Validace checklistu

Přímo před samotným kreslením procesů se provádí kontrola výběru elementů



Obrázek 7.5: Modelování procesu

nutných pro účel mapování předem definovaného procesu. Ve chvíli, kdy uživatel dokončí odškrtávání elementů z checklistů, stiskne tlačítko „Unlock“.

Aplikace ověří, jestli uživatelem zvolené prvky odpovídají správným prvkům pro tento proces. V případě úspěšného výsledku kontroly aplikace pogratuluje uživateli a umožní mu přistoupit k modelování procesů. V opačném případě upozorní uživatele na chybu.

Kontrola přítomnosti nutných propojení

První krok je kontrola přítomnosti propojení prvků. Ve chvíli, kdy bude uživatel chtít zkontrolovat stav svého nakresleného procesu, aplikace ověří, zda diagram obsahuje všechna nutná spojení. V případě selhání kontroly uživatel dostane chybové hlášení, ve kterém budou definované typy BPMN prvků, mezi nimiž chybí spojení. Stejná kontrola se provádí ve chvíli, kdy uživatel využívá nápovědu. Rozdíl je ale v tom, že v případě nálezu chyby během nápovědy uživatel dostane upozornění, které bude obsahovat název konkrétních elementů, mezi nimiž chybí spojení, nejen jejich typ.

Kontrola swimlanes

Druhý krok je kontrola swimlanes, která ověří, zda se všechny BPMN prvky nachází ve správných swimlanes. Správné umístění elementů je důležité zkontrolovat, aby se dodrželo BPMN pravidlo přiřazení činnosti správným rolím. Pokud kontrola selže, uživatel dostane chybové hlášení, která bude hlásit chybu ve swimlanes. V případě využití nápovědy se provede stejná kontrola s tím, že místo chybového hlášení uživatel dostane upozornění, které mu napoví, který element je špatně umístěn.

Kontrola počtu prvků na diagramu

Třetí a poslední krok je kontrola správného počtu prvků na diagramu. Simulátor zkontroluje, zda na diagramu nejsou žádné prvky navíc a zároveň zda jsou přítomny všechny nutné prvky. To znamená, že uživatel by neměl přidávat ani mazat žádné BPMN elementy kromě původních, které dostal k dispozici

při otevření kreslicího nástroje. Pokud při kontrole jedna z těchto podmínek selže, uživatel buď dostane chybové hlášení o chybném počtu elementů, nebo v případě využití nápovědy dostane upozornění na nutnou kontrolu počtu elementů na diagramu.

Kapitola 8

Výběr procesů a definování slovního popisu

V této kapitole se věnuji datům, které jsou obsažené v aplikaci. Krátce uvádím, jakým způsobem jsem vybíral procesy a slovní popis, které uživatel bude zpracovávat podle souvisejících úkolů jednotlivých fáze.

Podle pokynů této práce jsem měl za úkol navrhnout několik reálných situací, které představují proces a je možné je převést do notace BPMN. V návaznosti na tyto pokyny a dříve stanovené cíle (viz „**Úvod**“) jsem se rozhodl na začátku definovat slovní popis procesů firmy, na jehož základě by uživatel byl schopen zpracovat logické úkoly aplikace (viz kapitola „**Logika chování a uživatelské rozhraní aplikace**“).

V podkapitole „**Univerzitní kurz**“ jsem zmiňoval zkušenosti, které jsem získal během absolvování předmětu Procesní řízení na ČVUT. Z důvodů použití kvalitních materiálu ve své aplikaci, jsem se rozhodl použít slovní popis procesů, který byl součástí náplně tohoto vysokoškolského předmětu. Za účelem dodržení postupného nárůst úrovně složitosti v aplikaci jsem musel tento slovní popis lehce upravit a adaptovat na aplikační logiku. Spolu s vedoucím práce jsem navrhnul následující slovní popis, který odpovídá definovaným požadavkům a je využit v prvních dvou fázích aplikace v následné formě:

„***Naše společnost řeší proces vypsání nové pracovní pozice a následný nábor nového pracovníka na tuto pozici. Jsme středně velký podnik, který se pohybuje v dynamicky se vyvíjejícím odvětví a pro své potřeby vyžaduje často a jednoduše vytvářet a obsazovat nové pracovní pozice. Současný stav tohoto procesu je velmi “kostrbatý”, nejednotný a komplikovaný. Z těchto důvodů jsme se tento problém rozhodli řešit pomocí aplikace***.

Jakýkoliv pracovník ve společnosti může vytvořit požadavek na novou pracovní pozici. Při specifikaci této pozice je nutné uvést následující důležité údaje – požadavky na kandidáta a podrobnosti, které se týkají vypsané pozice. Po zadání těchto údajů je požadavek odeslán na oddělení HR (Human Resources), kde proběhne zpracování a kontrola správnosti údajů. Pokud odpovědný pracovník schválí, je pozice zařazena do databáze. V opačném případě je návrh zamítnut a žádající zaměstnanec je o zamítnutí informován e-mailem.

Po uložení pracovní pozice do databáze je spuštěn nový proces přijímání uchazečů na danou pozici, který začíná příchodem nové přihlášky od uchazeče

na pozici. Podávání přihlášek je řešeno pomocí jednotného formuláře. Po otevření tohoto formuláře jsou z databáze nejprve zobrazeny aktuální volné pozice ve společnosti. V tomto formuláři si uchazeč zvolí pozici, o kterou má zájem a následně vyplní do strukturovaného formuláře osobní údaje (vzdělání, pracovní zkušenosti, ...) a tyto údaje odešle.

Po příjmu této žádosti je automaticky provedena kontrola uchazeče na základě poskytnutých údajů a připojení na webovou službu Policie ČR, kde se nachází trestní rejstřík. Pokud služba vrátí informaci, že se uchazeč nachází v rejstříku trestů, je automaticky odmítnut a na zamítnutí upozorněn e-mailem. V opačném případě je přidán do seznamu potenciálních kandidátů na pozici.

Následuje manuální výběr nejlepších (top) uchazečů HR oddělením. Vybraní kandidáti jsou přizváni k pohovorům, během kterých si zaměstnanci HR mohou k jednotlivým uchazečům dělat poznámky. Po dokončení všech pohovorů je pracovníkem HR proveden finální výběr, po kterém se vybraný uchazeč stane zaměstnancem společnosti. O tom, že byl uchazeč vybrán je informován e-mailem. Stejným způsobem (e-mailem) jsou informování neúspěšní uchazeči o zamítavém stanovisku.“

Z výše uvedeného popisu lze pochopit, že popis obsahuje jeden velký proces, jež se člení do několika menších podprocesů. Za účelem naplnění cílů fáze modelování (viz kapitola „**Logika chování a uživatelské rozhraní aplikace**“) bylo potřeba tyto podprocesy převést ze slovního popisu do podoby modelu v notaci BPMN. Následně jsem navrhnul dva procesní modely, které jsou využité v aplikaci. Správnost a smysluplnost těchto modelů byla konzultována s odborným pracovníkem z oblasti procesního řízení – BPM architektem. Správná verze procesních modelů využitých v aplikaci je uložená v příloze „**Modely pro fázi modelování**“ ve formě obrázku.

Kapitola 9
Vývoj

V této kapitole podrobněji popíšu použité technologie a jejich využití ve své aplikaci, ukážu základní technické detaily, popíšu práci s knihovnou BPMN.io a s tím spojené činnosti. Kromě toho uvedu možnosti budoucnosti nástroje, jeho případného rozšíření a dalšího rozvoje.

9.1 Práce s frameworkem Vue.js

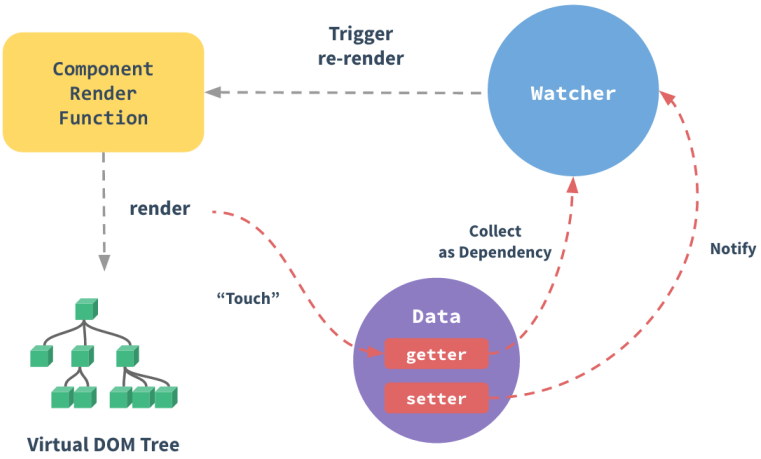
Po volbě frameworku Vue.js pro účel zjednodušení vývoje jsem se s ním musel nejdřív seznámit. V této podkapitole bych chtěl uvést myšlenky frameworku Vue.js, které se mi jevily zajímavé a užitečné. Tyto myšlenky jsem využil ve své aplikaci, což podpořím popisem aplikace z technického pohledu.

9.1.1 Pojem „reaktivita“

Začal bych pojmem reaktivita, na kterém se staví fungování jakékoli Vue aplikace. Pojem reaktivita znamená, že data uvnitř aplikace přímo souvisí s daty na obrazovce a jejich změna v kterékoli části okamžitě ovlivní překreslení zobrazení obrazovky. Ve frameworku React.js je reaktivita implementována takovým způsobem, že všechna data, která uživatel v aplikaci používá, jsou uložena ve state a props. Pokud je nutné změnit data, můžeme je změnit prostřednictvím metody „setState”. Dále React určí, které části aplikace jsou závislé na změněných datech, a překreslí je.

Vue.js používá podobný přístup, ale má jeden zásadní rozdíl; každé pole vstupních dat aplikace se rozšiřuje pomocí Object.defineProperty a je rozděleno do dvojic setter / getter. S jejich pomocí Vue sleduje, která data byla čtena nebo změněna, a může konkrétně určit, co ovlivňuje vykreslování zobrazení. Reaktivní přístup Vue.js je znázorněn na obrázku 9.1.

Cílem reaktivního přístupu je především okamžitá změna komponenty. V následující podkapitole popisuji komponentní přístup, jaké místo zaujímají komponenty ve frameworku Vue, jejich využití a funkčnost.



Obrázek 9.1: Reaktivní přístup Vue.js [8]

9.1.2 Vue komponenta a její struktura

Komponentní přístup

Komponentní přístup umožňuje vyhnout se nashromažďování kódu a jasně definovat architekturu aplikace. Každou komplexní stránku lze vždy rozdělit na menší komponenty. Každá z těchto částí, pokud je přiřazena ke komponentě, se snadněji udržuje, a v případě potřeby opakuje rozdělení uvnitř komponenty na ještě menší části.

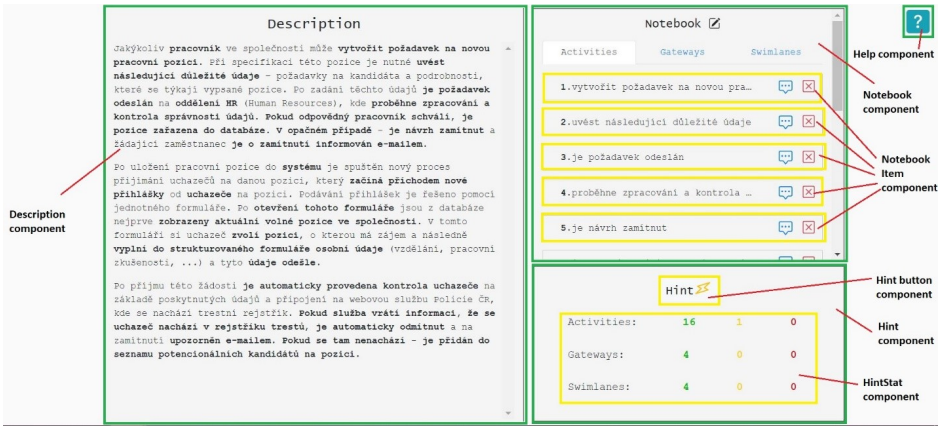
Velkou výhodou takového rozdělení na komponenty je snadná podpora a aktualizace aplikace díky tomu, že není potřeba pamatovat si logiku celé aplikace, a tak je možné se zaměřit na její konkrétní části. Jakékoli změny musejí být provedené pouze v jednom místě. Díky vlastnosti izolace lze zabránit konfliktům s jinými částmi aplikace. Pro ilustraci komponentního přístupu jsem uvedl obrázek 9.2, který zobrazuje rozpad aplikace na komponenty ve fázi analýzy slovního popisu (podrobněji uvedena v podkapitole „**Fáze 2 - analýza slovního popisu**“). Název komponent a případně jejich podřízených komponent je znázorněn přímo na obrázku. Nadřízené komponenty jsou zobrazené zelenou barvou, podřízené žlutou.

Struktura a organizace Vue komponenty

Pro snadnou organizaci komponentů a jejich součástí je navrženo použití Vue souborů. Tento soubor se skládá ze tří částí:

- <template> pro šablonu rozložení komponenty
- <script> s logikou komponenty
- <style> se stylem komponenty

Komponenty jsou stahovány pomocí nástroje vue-loader pro webpack, který skrývá složitost nasázení, což umožní používat technologie přímo z krabice



Obrázek 9.2: Rozpad fáze analýzy slovního popisu na dílčí komponenty

(třeba Sass, Bootstrap, CSS modules) a to bez jakéhokoli nastavení. Takovým způsobem vue-loader velmi usnadňuje používání externího zdroje.

Registrace a volání komponent

Vzhledem k tomu, že komponenty jsou nezávislé, v určitém okamžiku je nutné „požádat“ jednu o existenci druhé. Lze to provést pomocí dvou způsobů, a to buď globálně, nebo lokálně. Globálně znamená, že komponenta je zaregistrována hned po připojení Vue, ale před vytvořením nové instance. V případě lokální registrace, komponenty importujeme do části <script> Vue souboru (viz část „**Struktura a organizace Vue komponenty**“) a přidáváme názvy těchto komponent do sekce components. V praxi byla pro mě užitečnějším způsobem lokální registrace. Tím pádem mám přehlednější strukturu souboru a soustředím se na strukturu konkrétní komponenty. Komponenta se jednoduše vyvolává v části <template> Vue souboru a to pomocí využití tagu, který obsahuje název předdefinované komponenty (viz obrázek 9.3).

Předávání dat do komponenty a sledování událostí

Některé komponenty nemohou fungovat samostatně. V praxi většina z nich vyžaduje vstupní data, která jsou následně využitá komponentami za účelem provádění vnitřní logiky, generování <template> části aplikace a případně odeslání části dat do dalších vnitřních komponent. Pro přenos dat je nutná vazba na pojmenovaný atribut pomocí modifikátoru „v-bind“. Kromě toho, aby podřízená komponenta mohla používat tato data, je potřeba v ní specifikovat, pod jakým názvem by měla být očekávána. Specifikuje se to ve vlastnosti props podřízené komponenty.

Důležitou a velmi užitečnou vlastností je možnost sledování události pomocí jednoduchého modifikátoru „v-on“ anebo „@“. Kromě toho podřízená komponenta umí předávat událost do nadřízené komponenty pomocí vyvolání „$emit“. Pro lepší pochopení toto znázorňuji krátkým úryvkem svého kódu na obrázku 9.3.

Na obrázku je ilustrováno, jak se komponenty vestavují ve fázi modelování, která je rozdělená do dvou částí (podrobněji je to popsáno v podkapitole

„**Fáze 3 – Modelování a spojené s tím činnosti**“). V první části se uživatel pokusí správně definovat checklist, který je spojen s vlastností „listIsDone“, kterou můžete vidět vedle modifikátoru „v-if“. V kódu je napsáno, že pokud checklist není splněný, na obrazovce musejí být renderovány komponenty „ProcessDescription“ a „ModelingNotes“. Jakmile komponenta „ModelingNotes“ zaslechne událost „unlock-modeler“ od své podřízené komponenty, zavolá se funkce „UnlockModeler“, která nastaví „listIsDone“ na „!listIsDone“, zobrazí modální okno s gratulací a pomocí vestavěné Flip-animace změní obrazovku. Ve Vue tuto lze animaci použít velmi jednoduše pomocí klíčového slova „transition“. Jako výsledek se komponenta „Modeler“ zobrazí na celou obrazovku. Kromě používání jednotlivých komponent je důležitou věcí navigace mezi

```
<transition name="fade">
  <div v-if="!listIsDone" class="col col-5" id="example">
    <div class="row my-left-side">
      <ProcessDescription class="col col-12"/>
      <ModelingNotes class="col col-12" @hint-alert="showHintAlert" @unlock-modeler="UnlockModeler"/>
    </div>
  </div>
</transition>
<Modeler class="col col-7 model-window disabled" id="modeler"/>
```

Obrázek 9.3: Sledování události a změna stavu obrazovky

nimi, kterou krátce popíšu v následující podkapitole a ukážu příklad jejího využití ve své aplikaci.

9.1.3 Směrování pomocí vue-router

Vue.js poskytuje funkci směrování, která umožňuje uživateli přepínat mezi stránkami bez update stránky. Provozování směrovacího systému splňuje speciální knihovna vue-router. Aby bylo možné tuto knihovnu používat je potřeba explicitně uvést její použití pomocí Vue.use() (viz obrázek 9.4). V praxi jsem

```
import Vue from 'vue'
import VueRouter from 'vue-router'

Vue.use(VueRouter);

const routes = [
  {
    name: 'pre-analyze',
    path: '/pre-analyze',
    component: () => import('../views/PreAnalyzeView.vue')
  },
```

Obrázek 9.4: Směrování pomocí vue-route

toto považoval za jednoduché a velmi užitečné. Aplikaci mám postavenou na třech hlavních stránkách: pre-analyze, analyze a model, které jsou naplněné různými komponenty a odpovídají různým fázím aplikace. Jednotlivé fáze jsou popsané v kapitole „Logika chování a uživatelské rozhraní aplikace“. Pomocí

využití vue-routeru ukazuju aplikaci, kde by se měly jednotlivé komponenty zobrazovat a vysvětluju navigaci mezi nimi. Pro ukázku použití přikládám úryvek svého kódu na obrázku 9.4.

9.1.4 Řízení stavu aplikace s využitím Vuex

Pro účel řízení stavu aplikace používá Vue.js směs patternu a knihovny – Vuex. Slouží jako centralizované datové úložiště, které je dostupné pro všechny komponenty aplikace. Vuex obsahuje několik důležitých pojmů: state, store, getters, mutations, actions a modules. Postupně budu tyto pojmy vysvětlovat a uvedu ukázky jejich použití ve své aplikaci.

Všechna data ve Vuex jsou uložená ve stavech (anglicky state). Komponenty aplikace mohou tato data přijímat a aktualizovat se v případě jakékoliv změny. Tato data lze prohlížet buď přímo pomocí store nebo pomocí getters. Store spravuje stav, který se mění pouze zevnitř. Kvůli tomu nemohou externí komponenty přímo měnit stav.

Gettery vypočítávají vlastnosti na základě stavu úložiště (viz příklad na obrázku 9.5). Jsou užitečné, pokud několik komponent potřebuje vypočítat stejnou věc na základě dat úložiště. S použitím getterů můžete definovat logiku v jednom místě, což pomůže zabránit napsání stejného kódu v různých komponentech.

```
getters: {
  getFilteredNotes: state => {
    if (state.filter === 'activities') {
      return state.notes.filter(t => t.chosenType === 'activities')
    }

    if (state.filter === 'pools') {
      return state.notes.filter(t => t.chosenType === 'pools')
    }

    if(state.filter === 'gateways') {
      return state.notes.filter(t => t.chosenType === 'gateways')
    }
  },
```

Obrázek 9.5: Ukázka použití getteru, který vrací seznam filtrovaných podle typu poznámek

Pro aktualizaci stavu je potřeba provést mutace (anglicky mutations). Mutace se nestarají o byznysovou logiku aplikace, jejich jediným účelem je aktualizace stavu (viz příklad na obrázku 9.6). Mutace je synchronní (aplikace musí počkat, až bude dokončená mutace). Aby bylo možné zajistit úplnou kontrolu aktualizace stavu, mutace by měly být jediným způsobem provedení změny.

```
mutations: {
  addNoteToActivities(state, note) {
    state.notes.push(note);
  },
  setError(state, error) {
    state.error = error
  },
```

Obrázek 9.6: Ukázka použití mutací v kódu – změna stavu poznámkového bloku a aktualizace chyby

Akce (anglicky actions) obsahují byznysovou logiku aplikace a vůbec se nestarají o přímou aktualizaci stavu. To je udělané kvůli tomu, že actions jsou asynchronní (aplikace může běžet dál, i když akce ještě není dokončena). To je užitečné v případě, kdy je například potřeba dostat data z určitého API. Akce se volají mutaci, které už přímo aktualizují stav. Pro ukázku použití akce přikládám úryvek svého kódu na obrázku 9.7. Tento kód dostává jako parametr objekt ze slovního popisu, který následně zpracuje, a pokud validace projde, zavolá mutaci z obrázku 9.6 a přidá ji do poznámkového bloku.

```
actions: {
  addToActivities({ commit, state}, item) {
    if (item) {
      const note = {id: item.id, title: item.title, chosenType: 'activities', actualType: item.actualType};
      if(state.notes.find(note => note.id === item.id)) {
        commit('setError', 'Item is already in your notebook!');
      } else
        commit('addNoteToActivities', note);
    } else {
      commit('setError', 'You should select only the bold text!');
    }
  },
```

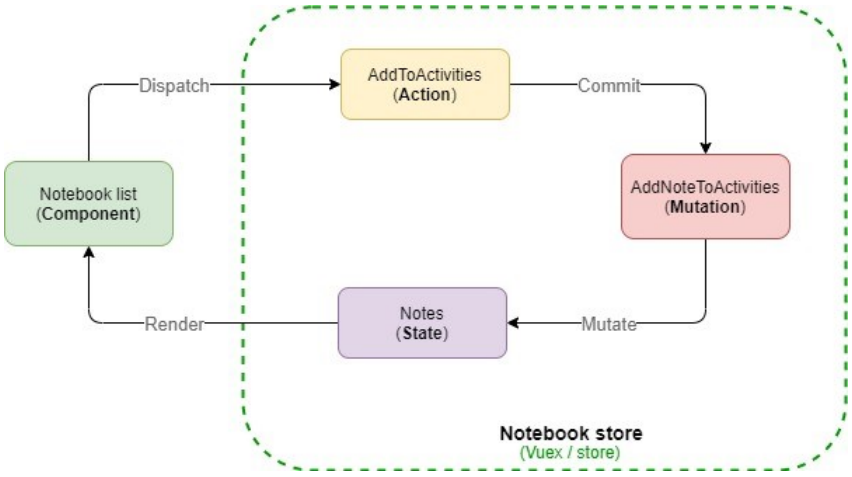
Obrázek 9.7: Ukázka použití actions v kódu

Poslední důležitý pojem ve Vuex jsou moduly. Jakýkoliv kus kódu lze dekomponovat, což znamená rozpad hlavního úložiště na několik samostatných modulů. Každý z těchto modulů ale musíte zaregistrovat.

Jsem přesvědčený, že v praxi se téměř žádná aplikace používající Vuex neobejde bez rozpadu jednoho velkého úložiště do několika drobných souborů neboli modulů. Ve své aplikaci jsem rozdělil store na moduly podle účelu jejich použití a podle souvislosti s byznysovou logikou komponenty určité fáze. Úložiště obsahuje jeden hlavní soubor, který do sebe importuje další moduly. Tím pádem modul notebook obsahuje state, ve kterém se budou ukládat jednotlivé BPMN prvky označené uživatelem. V části getters jsou představené filtry poznámek podle jejich typu (příklad na obrázku 9.5) a počítá se zde množství správných, špatných a chybějících poznámek (princip rozdělení je podrobněji popsán v podkapitole „**Fáze 2 - analýza slovního popisu**“). Mutaci tohoto modulu mění stav state, což znamená buď přidání nového prvku do úložiště (viz obrázek 9.6), smazání již zařazeného prvku, nastavení aktuálního filtru poznámek nebo případné nastavení chyby. V části

akce je definována logika, ve které je uvedeno, v jakou chvíli a za jakých podmínek se volají mutace úložiště poznámkového bloku (příklad na obrázku 9.7).

Pro lepší pochopení jsem se rozhodl zobrazit pomocí diagramu princip fungování Vuex na jednoduchém příkladě přidání BPMN prvků ze slovního popisu do poznámkového bloku uživatele. Je to ilustrováno na obrázku 9.8.



Obrázek 9.8: Proces přidání BPMN prvků do poznámkového bloku prostřednictvím Vuex

Uživatel označí ve fázi analýzy slovního popisu jakýkoliv text a následně zvolí typ BPMN prvků z context menu. V tuto chvíli se vyvolá akce (viz obrázek 9.7), která provede validaci a zavolá mutaci (viz obrázek 9.6), která následně změní stav poznámkového bloku. Jakmile se změní stav, uživatel uvidí aktualizace na své obrazovce. V závislosti na situaci se uživateli buď zobrazí chybové hlášení, anebo se zobrazí nový prvek v poznámkovém bloku uživatel.

9.2 Interakce s uživatelem

Na základě logiky aplikace, která je definována v kapitole „**Logika chování a uživatelské rozhraní**“ můžeme tvrdit, že aplikace má různé zdroje vstupních dat, které se liší v závislosti na fázi a účelu jejich použití. Po jejím zpracování by aplikace měla vrátit výsledek uživateli, aby tak mohl dostat tzv. zpětnou vazbu od systému. Tím pádem se vytváří komunikace mezi uživatelem a simulátorem, na jejímž základě se uživatel orientuje v aplikaci a rozhoduje, jakým směrem by měl postupovat. Výsledky zpracování a ověření jsou zobrazené dvěma způsoby: pomocí komponenty alert anebo pomocí modálního okna. Obsah těchto elementů záleží na fázi a konkrétním stavu aplikace.

Komponenta alert se používá buď na upozornění uživatele nebo na nahlášení nějaké chyby. Chybový alert se objeví ve chvíli jakékoliv kontroly v aplikaci a se červeně. Upozorňující alert se zobrazí v případech jakékoli nápovědy ze strany aplikace a má žlutou barvou.

Komponenta modální okno se zobrazuje při gratulaci uživateli, zobrazení tutoriálu nebo využití helpu. V případě gratulace uživateli je účelem této komponenty zobrazení hlášení o úspěchu a nasměrování uživatele do další části aplikace. Při zobrazení tutoriálu a helpu aplikace dostává uživatel nápovědu k tomu, jak má správně postupovat v určité fázi aplikaci. Příklad je ilustrován na obrázku 9.9, kde se modální okno zobrazí automaticky, když uživatel správně definuje všechny BPMN prvky ve fázi analýzy slovního popisu.

Obrázek 9.9: Příklad zobrazení modálního okna

9.3 Implementace modelování procesů

V této kapitole se zaměřím na část aplikace, která je zodpovědná za modelování procesů. Popíšu, jakým způsobem jsem umožnil uživateli převádět slovní popis procesů do podoby modelu v notaci BPMN, jaké jsem měl zkušenosti s knihovnou BPMN.io a uvedu validaci nakreslených diagramů.

9.3.1 Využití knihovny BPMN.io

Abych umožnil uživateli vytvářet BPMN modely a následně je testovat, musel jsem použít nástroj společnosti Camunda, který se nazývá BPMN.io. Je to knihovna, která poskytuje možnosti prohlížení a vytváření diagramů v notaci BPMN. Na rozdíl od jiných nástrojů pro modelování je knihovna BPMN.io open-source, což znamená že může být využitá kýmkoli a případně rozšířená podle cílů vývojáře. Kromě toho je napsána v JavaScript a nevyžaduje serverovou část, což umožňuje jednoduše ji integrovat do jakékoliv webové aplikace.

Webové rozhraní BPMN.io pro modelování se skládá z canvasu, což je oblast kreslení, sady nástrojů a několika funkcí pro podporu tvorby modelu. Panel nástrojů se skládá ze 13 prvků, které lze použít k interakci s kreslicí plochou. Celá knihovna je postavena na vykreslovacím nástroji (anglicky rendering toolkit) a webovém modeláři (web-modeler) bpmn-js. BPMN.io je navržena takovým způsobem, aby mohla poskytovat funkci prohlížeče (anglický viewer) a modeláře (anglický modeler). Cílem vieweru je načíst diagram v notaci BPMN 2.0 do aplikace a přidat mu vlastní data, zatímco modeler se používá k vytváření diagramů uvnitř aplikace. Pro realizaci těchto konceptů je bpmn-js postavená na dvou dalších důležitých knihovnách: diagram.js a bpmn-moddle.

Knihovna diagram-js je zodpovědná za rendering a modelování a obsahuje řadu základních služeb neboli servisů. Tyto servisy jsou v praxi funkce nebo instance, které mohou být využité v aplikaci pro změny stavu diagramu. Základní servisy jsou Canvas, EventBus, ElementFactory, ElementRegistry a GraphicFactory.

Ve své práci jsem ale našel využití hlavně pro Canvas a ElementRegistry. Canvas se používá pro účel přidávání a odebírání grafických prvků, zabývá se životním cyklem prvku a poskytuje API pro zoom a scroll obrazovky. Ten se používá pří načítání původního diagramu. Jak je to implementováno, je vidět na obrázku 9.10.

```
openDiagram(xml) {
  const self = this;
  self.modeler.importXML(xml, bpmnDiagram: function (err) {
    if(err) {
      self.handleError( err: 'failed to import diagram');
    } else {
      self.diagramXML = xml;
      self.handleShown();

      self.canvas.zoom('fit-viewport');
    }
  })
},
```

Obrázek 9.10: Využití servisu Canvas v aplikaci

Servis ElementRegistry obsahuje aktuální informaci o všech prvcích, přidaných do diagramu a poskytuje API pro načtení prvků a jejich grafické znázornění interním id. V praxi to znamená, že pomocí ElementRegistry knihovna BPMN.io poskytuje přístup k prvkům modelu. Pro využití tohoto servisu musí být nejdřív vytvořen nový objekt modeleru (viz obrázek 9.10). Tento objekt je kořenem modelovacího nástroje, řídí model na canvasu a všechny interakce s ním. Pro účel využití jakéhokoli servisu je potřeba tento servis u modelera vyžádat, jak je znázorněno na obrázku 9.11.

```
componentIsMounted () {
  this.modeler = new BpmnModeler({
    container: '#bpmn-canvas',
    keyboard: {
      bindTo: window
    }
  });

  this.elementRegistry = this.modeler.get('elementRegistry');
  this.modeling = this.modeler.get('modeling');
  this.canvas = this.modeler.get('canvas');

  this.createNewDiagram();
},
```

Obrázek 9.11: Vytvoření objektu modeler a příslušných servisů

Funkce, které poskytuje servis ElementRegistry jsou popsané v následující podkapitole, ve které se již budu zabývat validací diagramů nakreslených uživatelem.

9.3.2 Ověření nakreslených diagramů

ElementRegistry má řadu metod, které lze využít během práce s daty diagramů:

- elementRegistry.filter(condition): vrací pole prvků, které splňují podmínku
- elementRegistry.get(id): vrací prvek odpovídající určitému id
- elementRegistry.getAll(): vrací seznam všech prvků modelu
- elementRegistry.getGraphics(element, boolean): vrací grafické znázornění určitého prvku

První tři funkce jsem využíval ve své aplikaci pro ověření procesů, které uživatel nakreslil. Aby byl nakreslený model konzistentní, kompletní a jednoznačný, a aby ho bylo možné považovat za správný, musí být úspěšné tři hlavní kontroly:

- Kontrola přítomnosti nutných spojení, která v sobě v podstatě zahrnuje úplnost diagramu a zkontroluje, jestli není nějaký BPMN prvek izolovaně umístěn na ploše
- Kontrola swimlanes, což znamená kontrola správného umístění prvků na canvasu
- Kontrola počtu prvků, která již ověří, jestli uživatel náhodou nepřidal prvky navíc a jestli žádné prvky nesmazal

Jejich logika je podrobně popsaná v kapitole „**Logika chování a uživatelské rozhraní aplikace**“, zaměřím se tedy pouze na technické zpracování ověření modelu a postupně popíšu každou kontrolu.

Navíc je důležité si uvědomit, že knihovna BPMN.io již poskytuje základní validaci dodržení pravidel BPMN notace (viz kapitola „**Modelovací notace BPMN**“), proto není potřeba některé kontroly provádět, mezi nimi je například kontrola izolované šipky. Kromě toho, jak jsem uváděl v kapitole „**Fáze 3 – Modelování a s ním spojené činnosti**“, na canvasu budou již předpřipravené BPMN prvky včetně jejich názvů a případně i popisů. Díky tomu odpadá nutnost kontroly integrity každého prvku.

Všechny validace modelu probíhají v JavaScript souboru „model_check“. Metody z tohoto souboru vyvolávám v komponentě „Modeler“ pomocí tzv. wrapperu, který následně předává metodám nutné parametry. Ve chvíli, kdy uživatel stiskne tlačítko „Check“, zavolá se metoda „getErrors(elementRegistry, diagramChains, elemCount)“. Uvnitř této metody se provádí jednotlivé kontroly, jejichž implementace je popsaná níže, a probíhá dynamická aktualizace seznamů chyb na základě výsledků těchto kontrol. Na obrázku 9.12 je uveden kód realizace této metody.

```
getErrors(elementRegistry, diagramChains, diagramLanes, elemCounter) {
  let errors = [];

  const checkChains = this.checkAllChains(diagramChains, elementRegistry);
  checkChains.forEach(el => {
    const err = 'Bad connection between: ' + el.sourceRef.split("_", 1) + ' and ' + el.targetRef.split("_", 1);
    errors.push(err);
  });

  const laneError = this.checkSwimlanes(diagramLanes, elementRegistry);
  if(laneError) {
    errors.push(laneError);
  }

  const quantityError = this.checkElemCount(elemCounter, elementRegistry);
  if(quantityError) {
    errors.push(quantityError);
  }

  if(errors.length > 0) {
    console.log(errors.toString());
    return errors[0].toString();
  } else {
    console.log('There are no errors.');
    return 'There are no errors! You have completed the modeling!';
  }
},
```

Obrázek 9.12: Implementace validační metody, která vrací seznam chyb

První provedená validace je **kontrola přítomnosti nutných spojení**. Správná spojení jsou definována v modulu úložiště aplikace pojmenovaném „diagrams“ formou seznamu dvojic, kde první proměnná je ID zdrojového BPMN prvku a druhá je ID cílového BPMN prvku. Samotná kontrola je realizovaná metodou „checkAllChains(diagramChains, elementRegistry)“, která se pro každou dvojici spojení ze seznamu pokouší najít stejnou dvojici v diagramu. Jedná se o stahování aktuálního stavu všech šipek neboli Sequence Flows diagramu pomocí volání metody „elementRegistry.filter(condition)“, kde podmínkou je typ BPMN elementu. Pokud nějaké spojení chybí, zapíše toto spojení do seznamu, který na konci kontroly vrátí jako výsledek. Následně metoda „getErrors(. . .)“ (viz obrázek 9.12) zkontroluje tento výsledek, a pokud není prázdný, aktualizuje seznam chyb. Jak jsem tuto validaci implementoval je vidět na obrázku 9.13.

Druhou validací je **kontrola swimlanes**, která má dost podobnou realizaci jako první kontrola. V modulu „diagrams“ je uveden seznam swimlanes, kde každý prvek obsahuje seznam ID BPMN elementů, které by v něm měly být nakreslené. Pomocí metody „checkSwimlanes(diagramLanes, elementRegistry)“ aplikace porovná, jestli aktuální stav swimlanes na diagramu odpovídá výše uvedenému seznamu, a v případě, že narazí na jakoukoliv chybu, zaktualizuje seznam chyb, k čemuž použije stejný princip jako při ověřování přítomnosti nutných spojení. Aktuální stav je výsledkem metody elementRegistry.filter(condition). Podmínkou je v tomto případě název swimlanu.

Třetí a poslední kontrola je **kontrola správného počtu prvků** na diagramu. Pro každý diagram mám předem definovaný počet prvků podle jejich typu, který by měl obsahovat. V této validaci jednoduše porovnávám, jestli se aktuální počet BPMN elementů na diagramu shoduje s předem definovaným počtem. Kód pro validaci správného počtu prvků je uveden na obrázku 9.14.

```
checkAllChains(diagramChains, elementRegistry) {
  let res = [];
  diagramChains.forEach(chain => {
    const check =  this.checkTheChain(chain, elementRegistry);
    if(check.sourceRef && check.targetRef) res.push(check);
  });
  return res;
},

checkTheChain(chain, elementRegistry) {
  let res = {sourceRef: null, targetRef: null};
  const flows = this.getFlows(elementRegistry);
  if(!flows.find(flow => (flow.businessObject.sourceRef.id === chain.sourceRef)
    && (flow.businessObject.targetRef.id === chain.targetRef)))
  {
    res.sourceRef = chain.sourceRef;
    res.targetRef = chain.targetRef;
  }

  return res;
},

getFlows(elementRegistry) {
  return elementRegistry.filter(el => el.type === ('bpmn:SequenceFlow'));
},
```

Obrázek 9.13: Kontrola přítomnosti nutných spojení

```
checkElemCount(counter, elementRegistry) {
  const tasks = elementRegistry.filter(el => el.type.includes('Task'));
  const events = elementRegistry.filter(el => (el.type.includes('Event')));
  const gateways = elementRegistry.filter(el => (el.type.includes('Gateway')));
  const lanes = elementRegistry.filter(el => (el.type.includes('Lane')));
  if(counter.tasks !== tasks.length || counter.events !== events.length
    || counter.gateways !== gateways.length || counter.lanes !== lanes.length)
  {
    return 'Wrong number of elements!';
  }
},
```

Obrázek 9.14: Validace správného počtu BPMN prvků

Kapitola 10

Testování aplikace

Tato kapitola je věnovaná testování aplikace. Na úvod obecně popíši průběh a členění testování. Následně se zaměřím na konkrétní oblast testování, krátce popíši zkušenosti vybrané skupiny uživatelů a vyhodnotím jejich zpětnou vazbu. Závěrem se zabývám shrnutím výsledků testování aplikace a zamyšlením nad její budoucností.

10.1 Úvod do testování

Za účelem dosažení užitečné zpětné vazby jsem se rozhodl provádět testování jenom na cílové skupině uživatelů, kteří se zajímají o procesní řízení a kteří mají alespoň základní znalosti z této oblasti (viz kapitola „**Analýza a návrh požadavku**“). Celkem se mi podařilo zapojit do testování pět lidí. Při jednotlivém testování jsem nechal uživatele pracovat samostatně přibližně třicet minut. V případě jakýchkoli dotazů jsem poskytoval krátkou konzultaci a poznamenával jsem si jejich připomínky. Následně jsem po dokončení všech úkolů kladl otázky týkající se jedné ze třech klíčových částí, které jsou popsané v následujícím odstavci.

Testování jsem nejdřív rozdělil do dvou částí – vnitřní a vnější, kde k první skupině patří kontrola uživatelského rozhraní aplikace, k druhé její funkčnost. Aby bylo možné provést testování z různých úhlů, snažil jsem se nahlédnout ještě hlouběji a rozdělil jsem průběh testování do třech skupin, které pokrývají klíčové oblasti aplikace:

- Uživatelské rozhraní (GUI)
- Funkčnost (Functional)
- Použitelnost (Usability)

V dalších podkapitolách se věnuju těmto jednotlivým částem testování. Před tím, než se pustím do jejich popisu, bych chtěl zmínit, že při vývoji aplikace jsem průběžně po každé její fázi samostatně prováděl testování třech dříve definovaných vlastností: GUI, funkčnost a použitelnost. Z toho důvodů jsem se snažil průběžně odchytit co nejvíc potenciálních nebo již aktuálních chyb podle jejich zaměření a oblasti, čímž jsem ovšem znatelně zpomalil vývoj.

10.2 Testování uživatelského rozhraní

Jelikož si většina uživatelů nejprve všímá vzhledu webových aplikací, jako první se k testování nabízí samotné uživatelské rozhraní. V této části jsem ověřoval, jestli reálné zkušenosti uživatelů odpovídají definovaným požadavkům na aplikaci. Abych se v tom mohl vyznat kontroloval jsem uživatelské rozhraní podle následujícího seznamu:

- Model a prvky – kontrola správného umístění HTML prvků a dodržení jejich polohy
- Obsah – kontrola pravopisných a gramatických chyb
- Škálovatelnost – ověření zachování vzhledu aplikace při změně rozměru okna
- Kurzor – kontrola vzhledu kurzoru ve vstupních polích na prvcích, na které je možné kliknout
- Záhlaví – mělo by se držet jednoho standardu
- Fonty – zkušenosti uživatele s rozměrem a stylem fontů
- Scrollování – ověření přítomnosti scrollování, kde je potřeba
- Kompatibilita mezi prohlížeči – stránky se mohou v různých prohlížečích lišit

V této části nastal problém při změně velikosti obrazovky. Uživatelé, kteří měli menší velikost obrazovky, zejména 14palcovou či menší, měli mírně narušený vzhled modelu, který neodpovídal kladeným požadavkům. Z toho důvodu jsem přidal do uživatelské příručky a tutoriálu poznámku, která napovídá zmenšit rozměr okna o 10 procent pro vlastníky počítače s menší obrazovkou. Celkově mě ale výsledky potěšily, a navíc jsem dostal několik příjemných vyjádření ohledně minimalistického barevného zpracování a vhodně zvoleného fontu.

10.3 Testování funkčnosti

V této časti jsem ověřoval funkčnost a interakci aplikace, které by měly odpovídat popisu aplikační logiky, která je podrobně popsaná v kapitole „**Logika chování a uživatelské rozhraní aplikace**“. Mezi klíčové logické prvky využité v aplikaci patří:

- Input formy – liší se podle fáze aplikace
- Tlačítka – liší se podle fáze aplikace
- Collapse – vertikální container, využívá se při nápovědě

- Contextní menu – volba typu a následná aktualizace poznámkového bloku
- Tabs – filtrování podle typu BPMN prvků v poznámkovém bloku
- Tooltip – nápověda při navedení kurzorů na prvek
- Selection – zpracování uživatelem označeného textu
- Alerty – hlášení chyby, upozornění nebo nápověda
- Modální okna – objeví se při gratulaci nebo využití helpu
- Modeller – nástroj pro modelování

Při průchodu aplikací testeři ověřili funkčnost, kvalitu a správnost zpracování každého logického prvku z tohoto seznamu. Většina z těchto elementů úspěšně splnila testování funkčnosti. Avšak chyby a nesrozumitelnosti se objevily v následujících bodech: selection, alert a modální okno. Dále postupně uvádím jejich popis.

Zásadní chybou byla úplná nefunkčnost metody „document.getSelection()“, s jejíž pomocí uživatel označuje text ve druhé fázi aplikace (viz podkapitola „**Fáze 2 – analýza slovního popisu**“) v prohlížeči Firefox. Při rešerši jsem došel k závěru, že to bylo způsobeno programátorskou chybou prohlížeče [37]. Řešením tohoto problému bylo napsání své vlastní metody „getSelectionText()“.

Nesrozumitelnost s hlášením alert spočívala v překrytí jednoho hlášení jiným při současném využití nápovědy a provedení validace. Řešením v tomto případě bylo změna CSS vlastnosti „position“ u alert prvků.

Další funkční nesrozumitelnost se týkala modálního okna. Ve chvíli, kdy uživatel klikne kamkoliv mimo modální okno, toto okno se zavře. Řešením bylo využití Vue modifikátoru „prevent“, který se používá pro zpracování události.

10.4 Testování použitelnosti

Po testování uživatelského rozhraní a funkčnosti následovalo testování použitelnosti neboli usability. Je to závěrečná ale velmi důležitá část, která tvoří celkový dojem uživatele z aplikace. V této části jsem se zaměřil na následující body:

- Splnění / nesplnění očekávání uživatelů
- Logika rozhraní
- Navigace mezi a uvnitř stránky
- Dynamické zpracování a rychlost aplikace
- Kvalita zpětné vazby

- Schopnost zrušit poslední akce
- Kvalita a informační obsah helpu
- Soulad s cíli aplikace

V této části testování jsem dostal připomínky a návrhy na možné zlepšení od každého testera. Jednou z často zmiňovaných témat bylo zlepšení logiky nápovědy, a to v každé fázi. Pro většinu uživatelů by bylo lepší používat inkrementální variantu nápovědy, během které se budou postupně zobrazovat správné varianty. Podrobně logiku nápovědy popisuji v kapitole „**Logika chování a uživatelské rozhraní aplikace**“. Další připomínkou a žádoucím zlepšením byla možnost dělat si poznámky přímo ve slovním popisu pomocí barevného označení.

Kromě toho jsem zaznamenal, že zpracování tutoriálu a helpu mohlo být uděláno přehledněji a stručněji. Avšak většině uživatelů se líbila kvalita zpracování zpětné vazby, což jsem považoval za překvapivé.

Další připomínky se týkaly nedostatku technologie drag-n-drop za účelem rychlé a jednoduché změny typu BPMN prvků v poznámkovém bloku (viz kapitola „**Logika chování a uživatelské rozhraní aplikace**“). S tím naprosto souhlasím. Dokonce jsem měl v původním plánu na tomto aspektu pracovat, ale bohužel jsem ho nestačil implementovat.

Některé stížnosti se vztahovaly ke způsobu přidání textu do poznámkového bloku, který někteří testeři vnímali jako nepřívětivý. Musím přiznat, že tato kritika mě nepříjemně překvapila, jelikož jsem byl přesvědčen, že jsem zvolil skutečně vhodný způsob přidávání textu, který se blížil reálnému zápisu poznámek. V tomto případě jsem se inspiroval popisovací fixou.

Ostatní komentáře testerů se týkaly již samotného modelovacího nástroje a jeho úprav podle konkrétních potřeb. Například jedna z připomínek byla k poskytnutí větší samostatnosti pro uživatele během kreslení procesů a zmenšení předpřípravy elementů. Rovněž jsem byl upozorněn, že by bylo velmi užitečné zobrazovat nápovědu přímo na diagramu pomocí barevného označování. Momentálně ale bohužel netuším, jak to lze navrhnout a implementovat.

Nicméně, nehledě na velké množství komentáře, připomínek a výtek, byl celkový dojem testerů velmi pozitivní a aplikace splnila očekávání uživatelů, z čehož jsem měl velkou radost.

10.5 Souhrn výsledků testování

Ve výsledku z celého testování bych chtěl zmínit, že to byla cenná a nezbytná zkušenost ve vývoji aplikace. Na začátku jsem rozdělil testování to třech různých oblastí – testování uživatelského rozhraní, funkčnosti a použitelnosti. Podle těchto oblastí jsem následně prováděl testování. Chyby a možné zlepšení se vyskytly v každé z těchto částí, přičemž se největší počet poznámek objevil při testování použitelnosti.

10.6 Budoucnost aplikace a její další rozvoj

Krátce bych se chtěl také zamyslet nad budoucností aplikace a jejím možným rozvojem. I když výsledná aplikace plně naplnila moje očekávání a splnila zadání bakalářské práce, na základě výsledků testování si myslím, že potřebuje ještě řadu vylepšení. Aby nástroj mohl být běžně používán ve výuce a byl užitečný, v nejbližší budoucnosti bych chtěl provést úpravy podle připomínek z testování a následně společně s vedoucím této práce rozmyslet další možné vylepšení podle konkrétních požadavků. Četných zlepšení v kvalitě aplikace lze dosáhnout i ze strany jednoduchosti aktualizace dat aplikace a její obnovitelnosti.

Závěr

Tato práce se zabývá problematikou procesního řízení, její výukou, modelováním procesů pomocí využití notace BPMN a s tím spojenými činnostmi. V rámci zadání byly definovány čtyři cíle.

Prvním cílem práce bylo seznámení se s oblastí procesního řízení a následným modelování procesů. Za tímto účelem jsem se v první kapitole zabýval pojmy proces a procesní řízení, definoval jsem životní cyklus procesu a základní aspekty oblasti procesního řízení. Následně jsem se ve druhé kapitole zaměřil na modelování procesů a jeho součásti.

Druhý cíl spočíval v prostudování notace BPMN a určení jejích klíčových hledisek. Kvůli tomu jsem se ve třetí sekci věnoval samotné modelovací notaci BPMN, kde jsem zmínil náhled do její historie, popsal základní grafické prvky a závěrem uvedl strukturu BPMN notace a její typy diagramu.

V souladu s třetím cílem jsem se ve čtvrté kapitole dotknul činnosti a dovedností procesního analytika, a následně jsem v páté kapitole rozebral již existující způsoby vzdělávání v této oblasti. Tím považují první tři cíle, které se týkají teoretické části práce, za splněné.

Čtvrtým cílem bylo na základě dřív provedené rešerše navrhnout a implementovat simulátor, který ověří schopnost uživatele převést slovní popis procesu do podoby modelu v notaci BPMN, čemuž jsem se věnoval v dalších kapitolách.

V šesté kapitole jsem navázal na celkovou teoretickou část a po provedení krátké analýzy jsem definoval funkční a nefunkční požadavky na aplikaci.

Následně jsem se v sedmé kapitole věnoval popisu logiky a uživatelského rozhraní aplikace, kde jsem rozebral strukturu aplikace, uvedl funkčnost jejích jednotlivých částí a popsal jsem uživatelské rozhraní.

V další kapitole jsem se zabýval výchozími daty, která jsou obsažená v aplikaci, a důvodům, proč jsem je zvolil. Tato data odpovídají reálným situacím, představují proces a je možné je převést do modelu v notaci BPMN.

Následná kapitola se již týkala objasnění podstatných a důležitých věcí, se kterými jsem se setkal při vývoji aplikace.

Funkčnost, GUI a použitelnost aplikace byly otestované formou uživatelského testování na cílové skupině uživatelů. Popis jednotlivých částí testování je znázorněn v desáté kapitole.

Na základě výsledků testování jsem formuloval možný rozvoj a budoucnost

aplikace, která spočívá v úpravě a zlepšení jednotlivých částí aplikace podle poznámek a připomínek testerů. V nejbližší budoucnosti je nutné domluvit se s vedoucím této práce na tom, jak by bylo možné aplikaci zahrnout do výukového procesu a jaké jsou s tím spojené požadavky.

Výsledkem práce je funkční aplikace, která formou simulace ověřuje schopnost uživatele identifikovat ve slovním popisu správné elementy notace BPMN a propojit je do modelu, který dodržuje specifikaci notace a reflektuje slovní popis. V případě nesouladů ve vytvořeném modelu slovního popisu nebo ve identifikaci BPMN prvků je uživatel upozorněn na chyby, nebo případně jen na chybějící části.

Hlavní přínos práce spatřuji ve velkém osobním pokroku v programovací činnosti a v získání nových znalostí v oblasti procesního řízení a modelování, které budu s velkou pravděpodobností moct využít v praxi v nejbližší budoucnosti.

Příloha A

Literatura

[1] Šimonová Stanislava. *Modelování procesů a dat pro zvyšování kvality.* Brno: Computer Press, 2008. ISBN 978-80-251-1987-7.

[2] Basl Josef a Blažíček Roman. *Podnikové informační systémy: podnik v informační společnosti, 3., aktualizované a doplněné vydání.* Praha: Grada Publishing, 2012. ISBN 978-80-247-4307-3.

[3] Lean-Management. What is plan-do-check-act-cycle? Dostupné z: `https://kanbanize.com/lean-management/improvement/what-is-pdca-cycle`, navštíveno 15.5.2020.

[4] Professional Development. What is the dmaic cycle? Dostupné z: `https://www.professionaldevelopment.ie/lean-six-sigma-dmaic-cycle`, navštíveno 25.4.2020.

[5] Training Course Material. Bpmn 2.0 bethmann. Dostupné z: `https://training-course-material.com/training/BPMN_2.0_Bethmann`, navštíveno 29.4.2020.

[6] OMG. Business process model and notation version 2.0.2. Dostupné z: `http://www.omg.org/spec/BPMN/2.0.2/PDF/`, navštíveno 24.4.2020.

[7] Maxconsilium. Bpmn 2.0 models (part 2) - choreography model and conversation model. Dostupné z: `http://blog.maxconsilium.com/2013/09/bpmn-20-models-part-2.html`, navštíveno 30.4.2020.

[8] Vue.js. What is vue.js? Dostupné z: `https://vuejs.org/v2/guide/`, navštíveno 5.5.2020.

[9] Hammer M. and Champy J. *Reengineering the Corporation: A Manifesto for Business Revolution.* Harper Collins, New York, 1993. ISBN 9780061808647.

[10] Vlček R. *Hodnota pro zákazníka.* Management Press, 2000. ISBN 9788072610686.

[11] Český Normalizační Institut. ČSN EN ISO 9001:2009. *Systémy managementu jakosti – Požadavky.* Praha: Český normalizační institut, 2009.

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

[13] Grasseová Monika. *Procesní řízení ve veřejném sektoru.* Brno: Computer Press, 2008. ISBN 978-80-251-1987-7.

[14] Fiala Josef a Jan Ministr. *Průvodce analýzou a modelováním procesů.* Ostrava: VŠB - Technická univerzita, 2003. ISBN 8024805006.

[15] Řepa Václav. *Podnikové procesy. Procesní řízení a modelování 2., aktualizované a rozšířené vydání.* Praha: Grada Publishing, 2007. ISBN 978-80-251-1987-7.

[16] Pande, Peter S.; Neuman, Robert P.; Cavanagh, Roland R. *Zavádíme metodu Six Sigma: aneb jakým způsobem dosahují renomované společnosti špičkové výkonnosti.* Brno: Twins Com., 2002. ISBN 80-238-9289-4.

[17] Garimella, K.; Lees, M.; Williams, B.: *BPM Basics for dummies.* Hoboken, NJ: Wiley Publishing, Inc., Software AG speciální vydání, 2008. ISBN 9780470285718.

[18] Truneček Jan. *Znalostní podnik ve znalostní společnosti. 1. vyd.* Praha: Professional Publishing, 2003. ISBN 80-864-1935-5.

[19] Polan Jan. *Virtuální příprava procesních analytiků.* Praha, 2019. Diplomová práce. ČVUT. Fakulta elektrotechnická. Vedoucí práce Pavel Naplava.

[20] Cienciala Jiří. *Procesně řízená organizace: tvorba, rozvoj a měřitelnost procesů.* Brno: Computer Press, 2011. ISBN 978-80-7431-044-7.

[21] Mendling J. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness.* Springer Berlin Heidelberg, 2008. ISBN 9783540892243.

[22] Recker J., Rosemann M., Indulska M., etc.: *Business process modeling: a comparative analysis. Journal of the Association for Information Systems.* Brno: Computer Press, 2009. volume 10, no.4.

[23] HSBI. Organization business process modeling. Dostupné z: `https://hsbi.hse.ru/articles/analiz-i-modelirovanie-biznes-protsessov-organizatsii-predpriyatiya/`, navštíveno 22.4.2020.

[24] KPMS. Business process modeling. Dostupné z: `https://www.kpms.ru/Automatization/BPM.htm`, navštíveno 25.4.2020.

[25] StudMe. Business process modeling. Dostupné z: `https://studme.org/62398/menedzhment/modelirovanie_biznes-protsessov`, navštíveno 21.4.2020.

[26] Popper, Karl R.: *Logika vědeckého bádání*. Praha, 1997. ISBN 80-86005-45-3.

[27] ZaranTech. Top 10 responsibilities of a business analyst. Dostupné z: `https://www.zarantech.com/blog/top-10-responsibilities-business-analyst/`, navštíveno 1.5.2020.

[28] Make Me Analyst. 5 core activities of data analysis. Dostupné z: `http://makemeanalyst.com/5-core-activities-data-analysis-epicycles-data-analysis/`, navštíveno 1.5.2020.

[29] Peters-Kuhlinger, Gabriele a John Friedel. *Komunikační a jiné „měkké"dovednosti: využijte svůj potenciál, rozviňte své soft skills a staňte se úspěšnějšími. 1. vyd.* Praha: Grada Publ., 2007. ISBN 978-80-247-2145-3.

[30] Bridging The Gap. What business analyst skills are important for a new ba? Dostupné z: `https://www.bridging-the-gap.com/business-analyst-skills-important/`, navštíveno 3.5.2020.

[31] The BA Doc. Top business analyst "soft"skills (and how to develop them). Dostupné z: `https://thebadoc.com/ba-articles/f/top-business-analyst-soft-skills-and-how-to-develop-them`, navštíveno 2.5.2020.

[32] Zippia. 10 skills all business analysts should possess. Dostupné z: `https://www.zippia.com/research/business-analyst-skills/`, navštíveno 7.5.2020.

[33] Lawson Blake. Just-in-time vs. just-in-case learning. Dostupné z: `https://lawsonblake.com/just-in-time-learning/`, navštíveno 2.5.2020.

[34] Hartmann Oskar. *Just Do It!* Moscow: Eksmo, 2019. ISBN 978-5-91-482053-1.

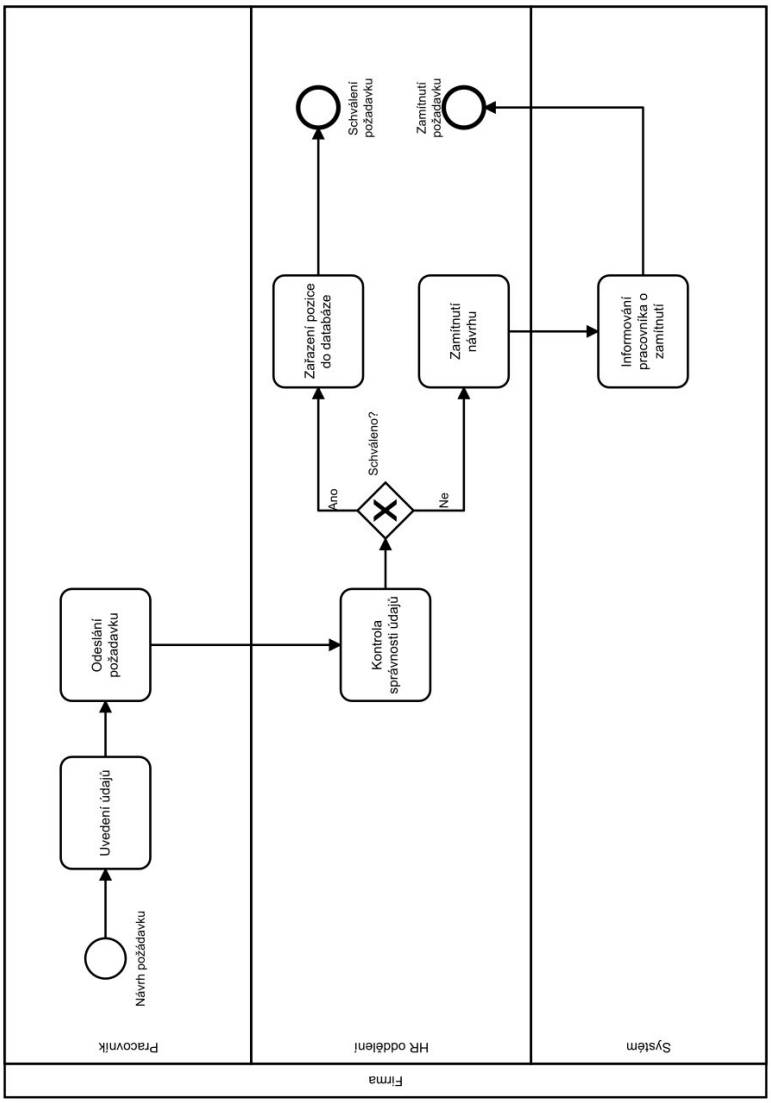
[35] Černý Michal. *Pedagogicko-psychologické otázky on-line vzdělávání*. Brno: Computer Press, 2009. volume 10, no.4.

[36] State of Js. Front-end frameworks – overview. Dostupné z: `https://2018.stateofjs.com/front-end-frameworks/overview/`, navštíveno 18.4.2020.

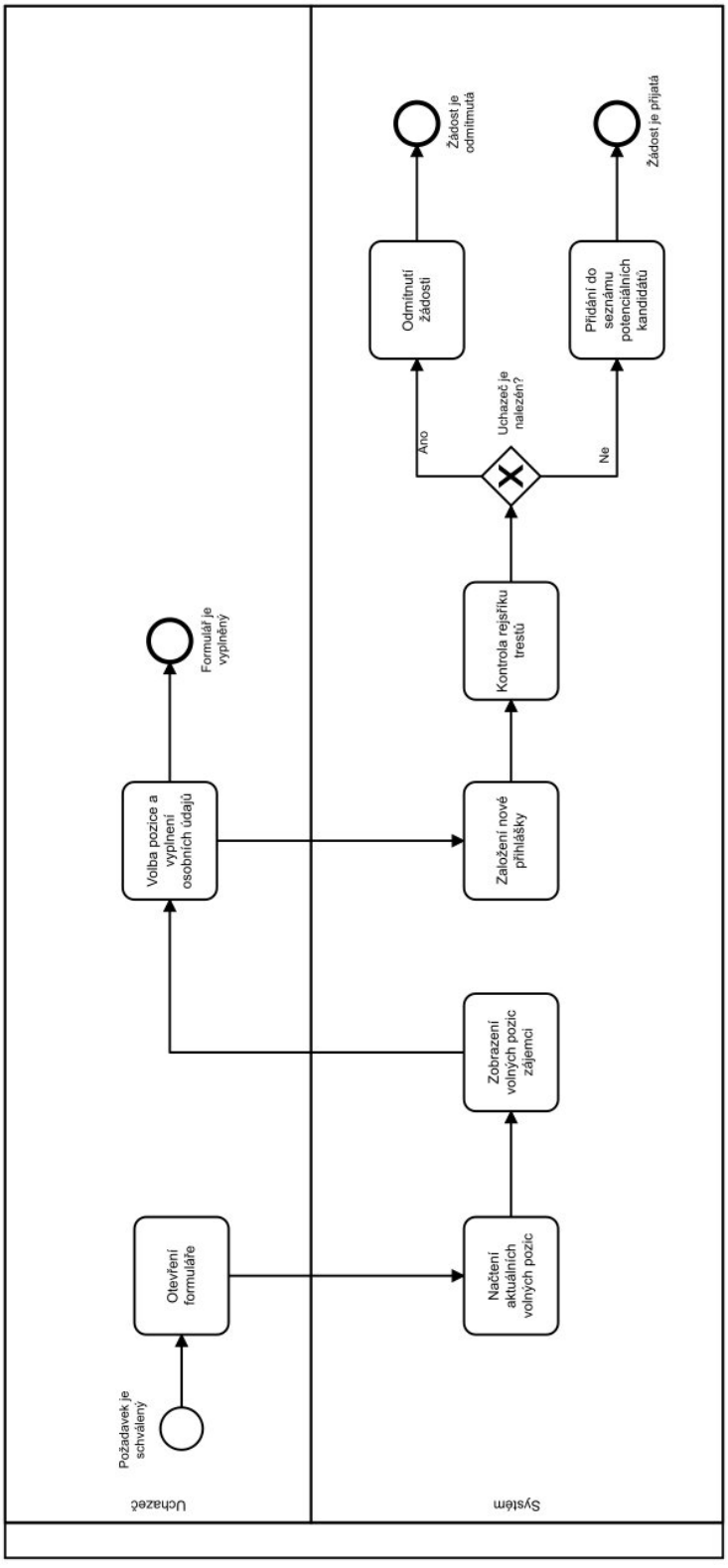
[37] Mozilla. Window.getselection() fails. Dostupné z: `https://bugzilla.mozilla.org/show_bug.cgi?id=85686`, navštíveno 10.5.2020.

Příloha B

Modely pro fázi modelování



Obrázek B.1: Model „Návrh požadavku“



Obrázek B.2: Model „Podání žádosti“

Příloha C

Obsah přiloženého CD

```
/
├── simulator-pm
│   ├── scr ..... zdrojové kódy implementace
│   │   ├── components
│   │   ├── router
│   │   ├── store
│   │   ├── utils
│   │   ├── views
│   │   ├── App.vue
│   │   └── main.js
│   └── config ..... konfigurační soubory
└── BP2020-zakhuvia.pdf ..... bakalářská práce ve formátu PDF
```