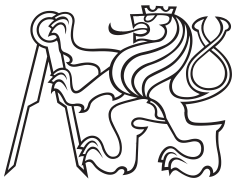
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Návrh a tvorba nástroje pro hodnocení kvality procesních modelů

Jan Zídek

Vedoucí: Ing. Radek Hronza
Studijní program: Otevřená informatika
Obor: Informatika a počítačové vědy
Květen 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jan Z í d e k

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Návrh a tvorba nástroje pro hodnocení kvality procesních modelů

Pokyny pro vypracování:

1. Seznamte se s dostupnými nástroji (respektive notacemi), určenými k modelování obchodních procesů.
2. Proveďte rešerši dostupné literatury a pro dané nástroje (respektive notace) vypište seznam existujících měr kvality procesních diagramů, pokud jsou pro ně vůbec definovány.
3. Proveďte analýzu nalezených měr kvality procesních diagramů a na základě výsledků analýzy definujte výsledný seznam měr vhodných pro notaci BPMN 2.0.
4. Z výsledného seznamu vyberte vhodné míry pro možnost automatizovaného výpočtu jejich hodnot, případně navrhněte nové míry na základě zkušeností získaných z výše uvedené rešerše.
5. Automatizovaný výpočet hodnot vybraných měr kvality procesních diagramů (vytvořených ve standardu BPMN 2.0) implementujte v prostředí Java.
6. Výsledkem bakalářské práce bude třívrstvá podniková aplikace, kde prezenční vrstva bude tvořena za pomoci dostupných webových frameworků. Ta bude s vrstvou obchodní logiky komunikovat pomocí REST webových služeb. Vrstva obchodní logiky bude komunikovat s datovou vrstvou dle uvážení (např. BaseX, objektová nebo relační databáze).
7. Výsledné řešení otestujte na dodaných vstupních datech, které dodá vedoucí bakalářské práce.

Seznam odborné literatury:

[1] Hronza, R., Pavlíček, J., Mach, R., & Náplava, P. (2015). Míry kvality v procesním modelování. Acta Informatica Pragensia, 4(1), 18-29. doi:10.18267/j.aip.57
[2] Hronza, R., Pavlíček, J., & Náplava, P. (2015). Míry kvality procesních modelů vytvořených v notaci BPMN. Acta Informatica Pragensia, 4(2), 140 - 153. doi:10.18267/j.aip.66
[3] Mach, R. (2015). Návrh a tvorba nástroje pro optimalizaci procesů na základě analýzy BPM modelů. Fakulta informačních technologií.
[4] OMG. (2014). Business Process Model & Notation (BPMN). Dostupné na: http://www.omg.org/bpmn/index.htm

Vedoucí bakalářské práce: Ing. Radek Hronza

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 24. 10. 2016

Poděkování

V první řadě bych chtěl poděkovat mému vedoucímu Ing. Radkovi Hronzovi, který i přes časové vytížení zodpovídal moje dorazy na požadavky nástroje a na problematiku měr kvality procesních modelů.

Nesmím též opomenout poděkovat mému kolegovi a kamarádovi Denisovi Baručićovi, se kterým jsem konzultoval některé možnosti struktur a objektových návrhů.

V neposlední řadě bych chtěl též poděkovat všem mým blízkým za jejich podporu a trpělivost při tvorbě této práce.

Prohlášení

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

V Praze, 22. května 2017

Jan Zídek

Abstrakt

Tato práce se zaměřuje na oblast výzkumu procesního modelování se zaměřením na míry kvality procesních modelů.

Úkolem této práce je navrhnout nástroj pro podporu tohoto probíhajícího výzkumu a pomoci tak automatizovat výpočty existujících měr kvality nad konkrétními procesními modely a také s pomocí při návrhu nových měr.

Nástroj je realizován jako vícevrstvá enterprise aplikaci na platformě Java EE s webovým rozhraním. Nástroj dokáže vypočítat známé míry kvality pro nahraný BPMN model. Administrátorům pak nabídne i možnost spojení známých měr kvality do nové komplexní míry kvality použitím matematického vzorce, kde mohou být použity známé míry kvality jako funkce. Nechybí ani možnost definice zpětné vazby na kvalitu modelu pro uživatele, která vychází právě z výsledků měr kvality.

Vzniklý nástroj je odrazovým můstkem k automatizované kontrole procesních modelů a byl vyvinut s ohledem na budoucí rozšíření o další možné funkcionality, které jsou v této práci též diskutovány.

Klíčová slova: nástroj, Java EE, míry kvality procesních modelů, procesní modelování, kvalita procesních modelů, BPMN, Business Process Management

Abstract

This thesis concentrate on the study field of business process modeling and quality measures of business process diagrams.

The task of this thesis is to design a tool to support this research. The tool is supposed to give opportunity of automatic evaluating of known quality measures for uploaded BPMN diagram. Also it should help with designing new quality measures.

The tool is realized as multilayered enterprise application based on Java EE platform with web user interface. The tool can process BPMN diagrams and count known quality measures for that diagrams. The admin users can also design a new quality measure as a mathematical expression which can contain known quality measure as a function. There is also an option to define user friendly feedback on quality of uploaded diagram. The feedback is based on results of quality measures.

Implemented tool the first step to automatic control of quality for business process diagrams. It was implemented as a core for many other possible features. These features are also discussed in this thesis.

Keywords: tool, Java EE, quality measures of process diagrams, process modeling, quality of process diagrams, BPMN, Business Process Management

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

1.1 Předmluva

Ve své bakalářské práci se zabývám kvalitou procesních modelů, resp. nástrojem na měření kvality procesních modelů. S modely je potřeba nějak pracovat, ne nadarmo se říká, že polovina úspěchu je problém pojmenovat. Budu se tedy dále ve své práci věnovat technologiím a standardům, resp. jak modely ukládat a jak s nimi pracovat. To je klíčové pro vytvoření výsledného nástroje pro počítání měr kvality.

Jednou z velmi důležitých částí mojí práce jsou míry kvality procesních modelů. Ty obvykle vyjadřují jednoduchost a hlavně srozumitelnost celého modelu. Moje práce neslouží primárně k nalezení nových měr, ale spíše jako odrazový můstek pro další výzkum v této oblasti. Výsledný nástroj implementuje již známé a ověřené míry kvality a nabízí tak rychlou zpětnou vazbu na konkrétní procesní model.

V další části práce se zabývám konkrétním návrhem nástroje pro počítání měr kvality procesních modelů. Zabývám se nejenom navržením nástroje z hlediska technologií, ale také z hlediska výsledné funkčnosti s ohledem na naplnění popisované vize.

1.2 Motivace a cíl

V dnešní době každá větší firma potřebuje optimální interní procesy, aby uspěla v konkurenčním boji. Proto je potřeba výzkum procesních modelů, které tyto procesy reprezentují, zvláště pak výzkum jejich optimalizace a kvality. O to se snaží výzkum měr kvality procesních modelů. Současný výzkum měr kvality probíhá ručním počítáním měr anebo formou dotazníků pro skupinu lidí. Tímto způsobem se výzkum snaží nalézat nové míry kvality, testovat stávající a zpřesňovat hodnoty jejich výstupů. Cílem této práce je celý proces zautomatizovat a dát nástroj vědeckým pracovníkům na počítání těchto měr přímo z procesních modelů pro zjednodušení a zrychlení výzkumu v této oblasti.

1.3 Struktura práce

Tato práce se skládá celkem ze 3 částí a 8 kapitol (včetně Úvodu).

1.3.1 Teoretická část

V první části se ve dvou kapitolách zabývám teorií potřebnou pro implementaci nástroje.

- V druhé kapitole rozebírám problematiku procesního řízení, resp. procesního modelování. Taktéž zde definuji, co to je kvalita procesního modelu a co jí ovlivňuje. V neposlední řadě se také zabývám standardy a jazyky, které slouží k uložení procesního modelu.
- V třetí kapitole představím problematiku měr kvality procesních modelů. Dále vymezím výčet měr kvality pro výsledný nástroj utříděné dle jejich kategorií.

1.3.2 Praktická část

V druhé části ve čtyřech kapitolách rozebírám nástroj samotný.

- Ve čtvrté kapitole se zabývám návrhem budoucího nástroje do vize, přes technologie a architekturu až po vymezení jádra nástroje, které zpracuji v rámci této práci.
- V páté kapitole představuji implementaci některých, algoritmicky zajímavých částí aplikace.
- V šesté kapitole pak nástroj probírám z uživatelského hlediska od instalace a spuštění až po ovládání přes grafické rozhraní.
- V sedmé kapitole popíši, jak probíhalo testování.

1.3.3 Závěr a přílohy

V poslední třetí části a osmé kapitole celou práci shrnu a zhodnotím.

- V přílohách je seznam použité literatury, seznam zkratek a další doplňující informace a obrázky.

Část I

Teoretická část

Kapitola 2

Business Process Management (Procesní řízení)

Business Process Management (zkráceně BPM) neboli česky Procesní řízení je velice rozsáhlá manažerská disciplína, která se zaměřuje například na plánování, sledování výkonnosti, mapování, analýzu a optimalizaci procesů, a další činnosti, které mají za cíl zlepšit fungování organizace.

Účelem této práce není se zabývat celou oblastí problematiky procesního řízení, proto se zaměřím pouze na její podmnožinu, kterou je právě procesní modelování. Do té spadá i výše zmíněná problematika měr kvality procesních modelů. Cyklus práce s procesy je znázorněn na obrázku 2.1.

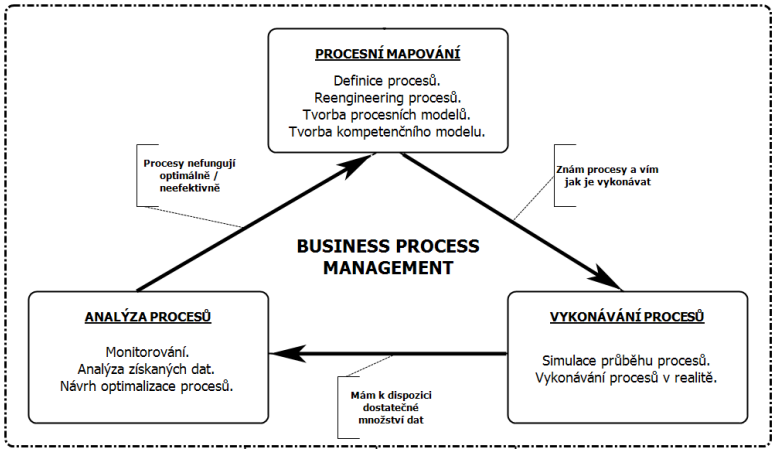
Více informací o celkové problematice procesního řízení je k nalezení například v knihách [5], [6] a [7].

2.1 Procesní modelování

Následující podkapitola má za úkol čtenáře seznámit s problematikou procesního modelování. Veškeré zde uvedené informace jsou kompilací informací ze zdrojů [1], [2], [5], [6] a [7].

Efektivita celé organizace spočívá v efektivitě jejích interních procesů, ať už administrativních, výrobních, obchodních, atd. Proto je zřejmé, že zvýšení efektivity interních procesů povede ke zvýšení efektivity celé organizace. Abychom mohli procesy optimalizovat, je třeba o nich něco vědět. K tomu slouží Business Process Management (zkráceně BPM), jehož cyklus práce s procesy je znázorněn na obrázku 2.1. Jak již bylo řečeno výše, součástí BPM je i procesní modelování, jehož výstupem je ucelená dokumentace zkoumaných procesů. To spočívá v seznámení se s principy dané organizace, monitoringu klíčových aspektů, příprava podkladů pro analýzu a podkladů pro zadávací dokumentaci pro vývojáře. Zjištěné poznatky lze znázornit procesními modely, o kterých pojednávám níže v podsekci 2.1.1 Procesní model.

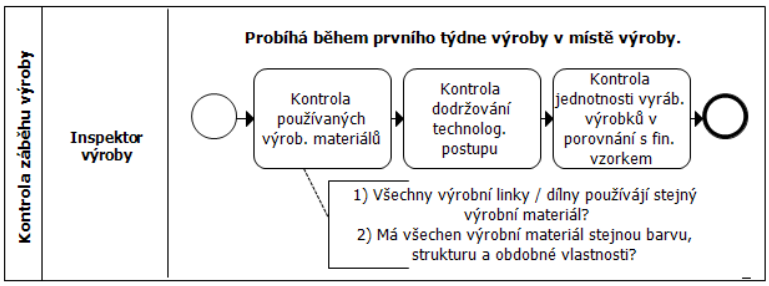
Na následujícím obrázku je znázorněn postup procesního modelování. Ve svojí práci se budu zabývat pouze oblastí procesního mapování, konkrétně tvorbou procesních modelu.



Obrázek 2.1: Cyklus Bussiness Process Management, zdroj: [1]

2.1.1 Procesní model

Procesní model je ucelená grafická forma popisující a znázorňující průběh daného business procesu. Jeho součástí jsou všechny části daného procesu a vztahy mezi nimi, popřípadě vztahy mezi jednotlivými, dílčími podprocesy. Slouží primárně k rychlému pochopení daného procesu, jeho aktivit a souvislostí s procesem spojených. Klíčové pro procesní model je jednoduchost, stručnost, jasnost a kvalita.



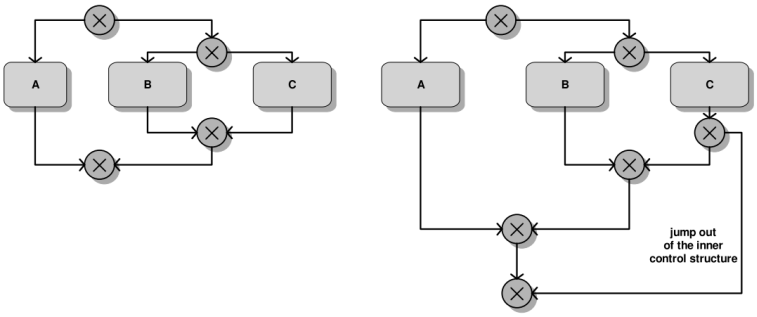
Obrázek 2.2: Príklad procesního modelu v notaci BPMN. zdroj: [1]

2.1.2 Kvalita procesního modelu

Dle článku [1] se kvalitou procesního modelu myslí převážně jeho jednoduchost, srozumitelnost a jasnost. Ta muže být například snížena subjektivním pohledem analytika nebo nevhodným použitím modelovacího nástroje.

Well-structuredness

Pojem „well-structuredness“ definoval van der Aalst [8] následovně: „Model je well-structured, pokud každá konstrukce split/join je správně vnořená.“ Jinými slovy, zda každý split má odpovídající join. Tento pojem tak definuje správné větvení modelu. Na následujícím obrázku je ukázka dobře (vlevo) a špatně (vpravo) větveného modelu.



Obrázek 2.3: Ukázka dobře (vlevo) a špatně (vpravo) větveného modelu, zdroj: [2]

Tento pojem uvádím spíše jen pro lepší představu o kvalitě modelu. Ve své práci se dále zaměřuji jen na modely v notaci BPMN, která dovoluje i struktury, které „well-structuredness“ více či méně porušují.

Způsoby, jak ovlivnit kvalitu modelu

Existuje několik způsobů, jak ovlivňovat kvalitu procesních modelů. Následuje výčet možných způsobů převzatých z článku [1].

- SEQUAL Framework, viz [9] a [10].
- The Guidelines of Modeling (GoM), viz [11].
- Quality Framework for conceptual modeling, viz [12] a ISO 9126.
- Seven Process Modeling Guidelines (7PMG), viz [13].
- Míry kvality procesních modelu, viz [1] a [14].

Obecně lze ovlivňování kvality rozdělit do dvou kategorií:

- Tvorba procesního modelu.
  - Je patrné, že kvalitu modelu lze ovlivnit už při samotném vzniku, kdy se analytik může řídit určitými doporučeními.
- Ověření kvality modelu a případná realizace úprav.
  - Jedná se o zpětnou vazbu k již vytvořenému modelu.

Ve své práci se budu zaměřovat pouze na míry kvality procesních modelů, které spadají do druhé kategorie, tedy zpětné vazby autorovi daného modelu. Míry kvality nabízí na rozdíl od ostatních způsobů ovlivňování kvality možnost automatizace, jelikož se jedná o matematický aparát, který dokáže kvalitu modelu reprezentovat číselnou hodnotou. Tím se liší od ostatních způsobů, které spíše definují obecná pravidla tvorby modelů, návrhové vzory, atd.

Dále bych rád zmínil, že každý kvalitní model by měl být tzv. „well-structured“. Tento pojem popisuji v podsekci 2.1.2.

2.2 Jazyky, standardy a nástroje pro tvorbu procesních modelů

Jazyků a standardů pro tvorbu procesních modelů je celá řada. V následujících několika větách bych rád představit několik z nich. Dále se pak budu podrobněji věnovat pouze třem nejpoužívanějším jazykům UML, XPDL a BPMN, jelikož ostatní zmíněné jazyky a standardy jsou již dnes zřídka používané a nemá tedy smysl o nich hovořit ani v kontextu modelovacích nástrojů, ani měr kvality procesních modelů.

2.2.1 Standardy pro práci s procesními modely

Vedle již tří zmíněných jazyků existují ještě další. Většina z nich je v dnešní době nepoužívaných nebo zřídka používaných. Jsou zde uvedeny hlavně pro úplnost a z historického významu. Následující výčet je převzat z článku [1].

- Unified Modeling Language (UML), zdroje [15, 16].
- XML Process Definition Language (XPDL), zdroje [17, 18].
- Business Process Model & Notation (BPMN), zdroje [19, 16].
- Event-driven Process Chain (EPC), zdroj [20].
- Finite State Machine (FSM), zdroj [21].
- Subject Oriented Business Process Management (S-BPM), zdroj [22].
- Yet Another Workflow Language (YAWL), zdroj [23].

2.2.2 Unified Modeling Language (UML)

Unified Modeling Language (UML) je grafický jazyk pro návrh, dokumentaci a vizualizaci. Používá se primárně pro objektový návrh softwaru nebo databázová schémata. Dá se však využít pro potřeby procesních modelů.

Jazyk je to velmi jednoduchý. Což může být pro určitý typ použití výhoda. Pro míry kvality procesních modelů je to však jazyk příliš obecný a je v tomto ohledu obtížně použitelný. Zvláště pak pro potřeby hlubší analýzy, automatizace a počítání měr kvality.

Modelovací nástroje pro jazyk UML

Zdarma dostupných i placených nástrojů pro jazyk UML je celá řada přes doplňky pro vývojová prostředí softwaru, webové aplikaci až po desktopové programy.

Za zmínku stojí například tyto modelovací nástroje:

- Zdarma dostupný doplněk pro vývojové prostředí NetBeans, easyUML[1].

[1] `http://plugins.netbeans.org/plugin/55435/easyuml`

- Desktopový nástroj Visual Paradigm, který je dostupný pro nekomerční účely zdarma[2]. Nabízí též integraci s vývojovými prostředími jako je například NetBeans, Eclipse, Visual Studio, IntelliJ IDEA a Android Studio.

- Online dostupný nástroj drow.io[3]. Tento nástroj je zdarma a nabízí plnou podporu modelování v jazyku UML. Možností je i uložení a nahrání rozpracovaného modelu z několika možných úložišť.

2.2.3 XML Process Definition Language (XPDL)

XML Process Definition Language (XPDL) je jazyk specializovaný na reprezentaci procesních modelů ve standardu XML. Je speciálně navržený tak, aby umožňoval zápis všech BPMN specifikací. V jednoduchosti by se dalo říci, že XPDL je jazyk navržený jako serializace BPMN procesních modelů.

V dnešní době je stále používaný a rozvíjený, nicméně není to tak rozšířený standard jako BPMN.

Modelovací nástroje pro jazyk XPDL

Jak bylo již zmíněno, XPDL slouží hlavně jako serializace BPMN procesních modelů. Nedá se tedy úplně hovořit o modelovacích nástrojích, ale o nástrojích, které dokáží importovat (exportovat) z (do) tohoto jazyka. Software v následujícím výčtu tedy umí pracovat s tímto jazykem.

- Desktopový Yaoqiang BPMN Editor[4] pro práci s BPMN procesními modely. Dokáže model jak z jazyka XPDL importovat, tak i do XPDL exportovat.

- Placený nástroj Together XPDL Workflow Editor.

- Desktopový Bizagi Process Modeler[5]. Dokáže model jak z jazyka XPDL importovat, tak i do XPDL exportovat.

2.2.4 Business Process Model & Notation (BPMN)

Business Process Model & Notation (BPMN) je dnes nejrozšířenějším standardem pro tvorbu procesních modelů. Jedná se o grafický jazyk založený na jazyku UML, definující navíc standardy specializující se na práci s procesními modely. BPMN dále definuje standardizaci pro uložení ve formátu XML.

Díky jeho specializaci na modely business procesů je tento standard vhodný pro další výzkum a zpracování v této oblasti. Například pomocí měr kvality procesních modelů. Díky tomuto a díky rozšířenosti nejenom ve světě, ale i

[2] https://www.visual-paradigm.com/download/community.jsp
[3] https://www.draw.io/
[4] http://xpdl.yaoqiang.org/
[5] http://www.bizagi.com/en/products/bpm-suite/modeler

na ČVUT (například v Procesním portále ČVUT[6]) budu v této práci dále pracovat pouze z modely v této notaci.

Více o Procesním portále ČVUT zde [24].

Modelovací nástroje pro jazyk BPMN

Kromě již zmíněných nástrojů pro standard XPDL, existují i nástroje, které pracují čistě se standardem nebo v notaci BPMN. Následuje výčet některých z nich.

- Desktopový i webový zdarma dostupný modelovací nástroj Camunda Modeler[7]. Pracuje s modely i je ukládá v notaci BPMN.
- Placený webový nástroj IBM Blueworks Live[8].
- Již zmíněný online dostupný nástroj drow.io[9] dokáže též vytvářet procesní modely standardu BPMN. Bohužel ale nedokáže tento model exportovat do notace BPMN. Nabízí sice možnost uložení ve formátu XML, je to ale bohužel jen serializace grafického modelu.
- Již zmíněný desktopový Bizagi Process Modeler[10].

2.3 Shrnutí kapitoly

V této kapitole jsem nastínil problematiku procesního řízení. S ohledem na rozsah a obsah práce, detailněji jsem se zabýval pouze oblastí procesního modelování. Z této oblasti jsem se věnoval tvorbou procesních modelů, resp. jejich kvalitou. Mimo jiné jsem představil problematiku kvality procesních modelů a možností jejich zaznamenávání, resp. jazyky a standardy pro jejich záznam.

[6] https://procesy.cvut.cz/
[7] https://camunda.org/download/modeler/
[8] https://www.blueworkslive.com/home
[9] https://www.draw.io/
[10] http://www.bizagi.com/en/products/bpm-suite/modeler

Kapitola 3

Míry kvality procesních modelů

Při rešerši dostupné literatury o mírách kvality procesních modelů jsem narazil na článek [14]. Ten se zabývá mírami kvality pro procesní modely obecně. Doplňující rešerší jsem našel již zmíněný článek [1] a závěrečné kvalifikační práce [2], [3], [25] a [26], které článek [14] doplňují a zabývají se mírami kvality pro notaci BPMN.

Kvalitou procesních modelů, resp. mírami kvality a uživatelským testováním se také zabývají zdroje [27] a [28].

Jak jsem již uvedl v podkapitole 2.2.4 o notaci BPMN, budu se dále zabývat pouze mírami pro tuto notaci. Po konzultaci s výzkumným týmem a vedoucím této bakalářské práce jsem došel k závěru, že míry kvality uvedené v článku [1] jsou pro výsledný nástroj v této chvíli dostatečné a následující výčet měr z něj vychází. Tento výčet je však ještě rozšířen o informace ze zdrojů [2], [3], [25] a [26].

Míry kvality obvykle vycházejí z měr kvality pro softwarový kód a dělí se do několika kategorií. Míry, které jsem vybral, spadají do kategorií „Complexity“, „Comprehensiveness“, „Modularity“, „Size“, a „Structure“. Těmito kategoriemi a vybranými mírami se zabývám v jednotlivých podkapitolách níže. Dělení měr kvality do těchto kategorií vychází z článku [1] a diplomových prací [2], [3] a [25].

3.1 Complexity

Míry kvality kategorie „complexity“ vyjadřují hlavně složitost modelu z hlediska jeho průchodu čtenářem.

3.1.1 Coeficient of Network Complexity

Tato míra vyjadřuje náročnost porozumění modelu.

$$CNC = \frac{E}{V}$$

$$E = počet\, hran$$

$$V = počet\, uzlů$$

3.1.2 Control Flow Complexity

Control Flow Complexity (česky „Složitost řídícího toku“) ve zkratce CFC je míra, která vyjadřuje počet lineárně nezávislých cest v procesu.

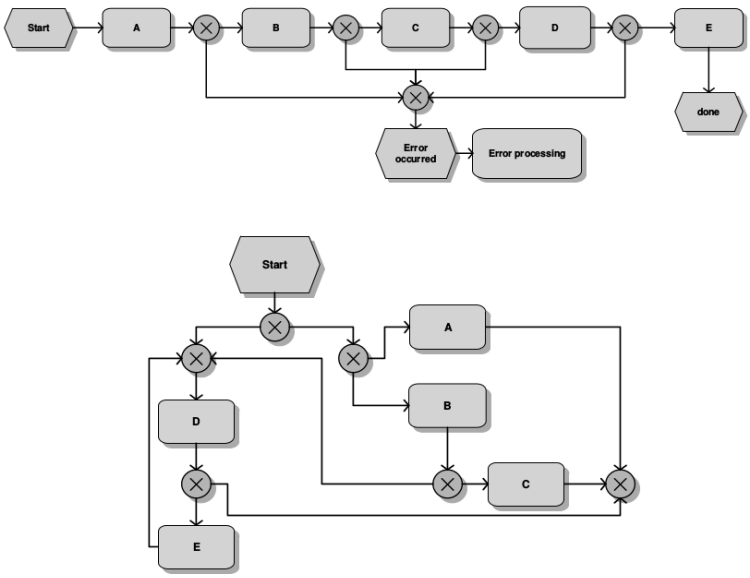
$$CFC(p) = \sum_{a \in p,\, a \equiv XOR-split} CFC_{XOR-split}(a) + \sum_{a \in p,\, a \equiv OR-split} CFC_{OR-split}(a) + \sum_{a \in p,\, a \equiv AND-split} CFC_{AND-split}(a)$$

$$CFC_{XOR-split}(a) = \text{Počet propojení vystupujících z daného exkluzivního } (XOR) \text{ rozhodovacího bloku}$$

$$CFC_{OR-split}(a) = 2^{(\text{počet propojení vystupujících z daného inkluzivního } (OR) \text{ rozhodovacího bloku})} - 1$$

$$CFC_{AND-split}(a) = 1$$

Čím je vyšší hodnota CFC, tím je vyšší složitost modelu. Ukázalo se, že je vysoká korelace hodnoty této míry s porozuměním modelu. Avšak jako nevýhodu je potřeba zmínit, že tato míra nijak nezohledňuje strukturu procesu. Jako demonstraci uvádím dva obrázky zobrazují dva procesy se stejným CFC. Je však zřejmé, že druhý model není „well-structured“. Zdroje: [2], [3] a [29].



Obrázek 3.1: Ukázka dvou modelů se stejným CFC, Zdroj: [3]

3.2 Comprehensiveness

Kategorie „comprehensiveness“ vyjadřuje srozumitelnost pro čtenáře daného modelu.

3.2.1 Cognitive weight

Tato míra vyjadřuje náročnost porozumění modelu. Na rozdíl od předchozí míry je přihlédnuto k různému vnímání jednotlivých objektů v modelu. Těm je přiřazena váha, která je získaná na základě empirického výzkumu. Celková srozumitelnost modelu je potom vážená suma všech objektů modelu.

$$CW(p) = \sum_{a \in p} W(a)$$

$$W(a) = váha\, elementu\, a$$

Jako nastavení těchto vah nechám váhy z následující tabulky, které vzešly z výzkumu Gruhn a Laue [29]. Tyto informace čerpám z diplomových prací [2] a [3].

BPMN řídící struktura	váha
Po sobě následující aktivity	1
Exkluzivní brána se dvěma vetvemi	2
Exkluzivní brána se třemi a více větvemi	3
Paralelní brána	4
Inkluzivní brána	7
Podproces	2
Běh více instancí aktivity	6
Zrušení jedné aktivity	1
Zrušení více aktivit	2 nebo 3

Tabulka 3.1: Kognitivní váhy dle výzkumu Gruhn a Laue. Tabulka je převzata z prací [2], [3].

Ve své práci však nebudu zohledňovat poslední tři řídící struktury a jejich váhy vynechám. Jedná se totiž o struktury spojené s elektronizací procesů, resp. se spouštěním a během modelů. Elektronizace však s touto prací přímo nesouvisí.

3.3 Modularity

Kategorie „modularity“ vyjadřuje modulárnost modelu neboli jak moc je využitý / jak moc využívá jiné zdroje. Dalo by se též říci, že vyjadřuje, jak moc je model provázaný z jinými procesy / zdroji.

3.3.1 Maximum depth

Tato míra vyjadřuje maximální hloubku zanoření podprocesů. Jinými slovy, v jaké hloubce se nachází nejvíce zanořený podproces. Hlubší zanoření podprocesů může vést na obtížnější porozumění celého modelu.

Pokud model obsahuje cyklus, tato míra vyjadřuje hloubku tohoto cyklu.

3.3.2 Mean depth

Tato míra vyjadřuje průměrnou hloubku zanoření podprocesů. Jinými slovy, v jaké hloubce se nachází průměrně zanořený podproces. Hlubší zanoření podprocesů může vést na obtížnější porozumění celého modelu.

Pokud model obsahuje cyklus, tato míra vyjadřuje hloubku tohoto cyklu.

3.3.3 Has cycle

Tato míra indikuje, zda se v modelu nachází cyklus na úrovni podprocesů.

$$HC = \begin{cases} 1 & model\,obsahuje\,cyklus \\ 0 & jinak \end{cases}$$

Tuto míru jsem navrhl z důvodu potřeby detekce cyklů v procesním modelu, který se skládá z více podprocesů. Právě kvůli cyklům se míry počítající s hloubkou modelu nedají jednoznačně určit.

3.4 Size

Míry kvality kategorie „size“ jsou hlavně statistického charakteru. Vypovídají hlavně o velikosti modelu. Odtud plyne i její název. Jedná se o nejjednodušší míry popisující počty různých prvků, vazeb, atd.

3.4.1 Number of Activities

Tato míra vyjadřuje počet všech prvků modelu typu aktivita/činnost.

$$NOA = počet\,aktivit\,v\,procesu$$

Pro zajímavost lze uvést, že pro softwarový kód by se dala použít podobná míra Lines of Code.

3.4.2 Number of Events

Tato míra vyjadřuje počet všech prvků modelu typu událost.

$$NOE = počet\,událostí\,v\,procesu$$

3.4.3 Number of Gateways

Tato míra vyjadřuje počet rozhodovacích bloků v modelu.

$$NOG = počet\,rozhodujících\,bloků\,XOR + počet\,rozhodujících\,bloků\,OR + počet\,rozhodujících\,bloků\,AND + počet\,rozhodujících\,bloků\,založených\,na\,událostech$$

3.4.4 Number of Data

Tato míra vyjadřuje počet objektů informačního typu.

$$NOD = počet\,dokumentů + počet\,externích\,skladů\,informací$$

3.4.5 Number of Flow Elements

Tato míra vyjadřuje počet objektů sloužících k vytvoření vazeb a propojení mezi jednotlivými prvky modelu.

$$NOFE = počet\,propojení\,v\,modelu$$

3.4.6 Další míry kategorie „Size“

Další míry z kategorie „size“ jsou opět jen počet prvků daného typu v modelu.

Všechny míry kvality implementované v tomto nástroji jsou k nalezení v příloze C.

3.5 Structure

Míry kvality typu „structure“ vyjadřují kvalitu návrhu procesního modelu z pohledu vnitřní struktury elementů.

3.5.1 Nesting depth

Nesting depth (česky „Hloubka rozhodovacího zanoření“) udává počet rozhodnutí, které jsou třeba vykonat v průběhu vykonávání procesu.

$$\begin{aligned} ND = {} & počet\,exkluzivních\,(XOR)\,rozhodovacích\,bloků + \\ & počet\,inkluzivních\,(OR)\,rozhodovacích\,bloků + \\ & počet\,rozhodovacích\,bloků\,založených\,na\,událostech \end{aligned}$$

3.5.2 Interface of Complexity

Míra Interface of Complexity (česky „Složitost rozhraní“) vyjadřuje složitost procesu z pohledu jeho datových vstupů a výstupů.

$$IoC = length + (param_{in} \cdot param_{out})^2$$

$$param_{in} = počet\,datových\,vstupů$$

$$param_{out} = počet\,datových\,výstupů$$

Délka (length) je definovaná jako počet aktivit v procesu. Pokud tento počet není znám, jinak řečeno se jedná o „black box“ je délka definovaná jako 1.

3.5.3 Multiple Use Of Decision Blocks In Direct Response

Míra Multiple Use Of Decision Blocks In Direct Response (česky „Násobné využití rozhodovacích bloků v přímé návaznosti“) vyjadřuje počet rozhodovacích bloků, které jsou použity v přímé návaznosti.

3.5.4 Number of Duplicities

Tato míra vyjadřuje, kolik prvků v modelu je potenciálně duplicitních. Často se stává, že vlivem nepozornosti analytika je v modelu nějaký element více krát.

V nástroji je rozpoznávání duplicit realizováno přes podobnost jmen elementů. Porovnávání jmen není case sensitive[1] a neberou se v potaz jiné znaky než písmena a čísla, tj. pomlčky, tečky, čárky a jiné jsou ignorovány.

3.6 Shrnutí kapitoly

V této kapitole jsem představil problematiku měr kvality procesních modelů. Dále jsem po konzultaci s výzkumným týmem a s vedoucím této práce sestavil výčet měr kvality, které budou v nástroji implementovány.

Některé z nich jsem v této kapitole blíže popsal. Všechny míry kvality implementované v tomto nástroji jsou k nalezení v příloze C.

[1] Při porovnávání se neberou v potaz malá a velká písmena.

Část II

Praktická část

Kapitola 4

Návrh nástroje pro měření kvality procesních modelů

V této kapitole se zabývám vizí výsledného nástroje, výběrem technologií, které použiji na implementaci nástroje, a architekturou budoucí aplikace. Hodlám se zaměřit i na budoucí vývoj aplikace mimo rozsah této práce. V neposlední řadě v této kapitole definuji, v jakém rozsahu nástroj implementuji v rámci této práce.

4.1 Vize nástroje

V této části nastíním vizi výsledného nástroje. Rád bych zdůraznil, že se zaměřuji i na funkcionality, které přesahují rozsah této práce a bude třeba ještě dalšího vývoje. Funkcionalitu implementovanou v rozsahu této práce vymezuji v sekci 4.4.

Nástroj implementuje výčet měr kvality procesních modelů uvedených v kapitole 3 a příloze C. Každá míra bude uživatele informovat o hodnotě jejího výsledku nad jím vybraným modelem. Zároveň bude uživatele informovat, co daná míra znamená a co znamená její výsledek. Velmi vítanou možností je sada doporučení, jak by uživatel mohl model vylepšit. Doporučení budou generována právě na základě výsledku dané míry kvality.

Každý uživatel bude moct vybrat model reprezentující proces včetně podprocesů (včetně podprocesů v externích souborech) a nechat si na něm spočítat míry kvality, které budou v nástroji definované. Uživatel bude moct tento model vybrat z listu jemu dostupných modelů. Ty mohou být uložené v databázi nebo získané z externích systémů, například z Procesního portálu ČVUT.

Administrátor může definovat nové zdroje procesních modelů, například jednotlivé instance již zmíněného Procesního portálu ČVUT. Z tohoto zdroje se pak v předem definovaném standardu budou stahovat modely.

Administrátor může upravovat informace o známých mírách a definovat míry nové. Nové míry mohou být definovány jako matematický výraz, kde již známé míry jsou reprezentovány jako matematické funkce. Definice nových měr má v tomto pohledu význam hlavně proto, že většina měr je statistického charakteru (prakticky všechny míry kvality typu Size). Takže se tyto míry

dají vhodně využít k definici nových komplexnějších měr kvality.

Jednotlivým mírám může administrátor měnit defaultní parametry (pokud nějaké vstupní parametry mají) a intervaly výsledků, které definují ohraničení výsledných hodnot míry. Tyto intervaly pak mohou být pro uživatele klíčovým směrodatným prvkem, jak kvalitní jeho model je a jak by mohl daný model vylepšit. Obojí může být reprezentováno nejenom číselným výsledkem, ale i slovním popisem v lidsky přijatelné podobě. Tato sada doporučení na základě výsledku míry kvality může být posílaná i do externích modelovacích nástrojů, například do již zmíněného Procesního portálu ČVUT.

Následuje výčet funkcionalit, který zohledňuje finální verzi nástroje přesahující rámec této práce.

4.1.1 Výčet funkčních požadavků na nástroj

- Každý uživatel bude moci provádět výpočty měr kvality nad jím vybraným modelem.
  - Model může vybrat z listu uložených / dostupných modelů.
  - Model může nahrát jako samostatný BPMN soubor / ZIP archiv obsahující několik BPMN souborů reprezentujících komplexní proces s několika podprocesy.
- Každý přihlášený uživatel může:
  - Upravit informace o jeho profilu a jeho přihlašovací údaje.
  - Uložit si nahraný procesní model pro budoucí použití.
  - Počítat míry kvality i na procesních modelech získaných z externích systémů.
- Administrátor může:
  - Definovat nové zdroje procesních modelů.
  - Upravovat informace o známých mírách.
  - Vytvářet nové míry na základě matematického výrazu, který může obsahovat ostatní míry jakožto matematické funkce.
  - Vytvářet nové a spravovat uživatele nástroje.
  - Spravovat intervaly hodnot měr, jakožto logické milníky jejích výsledků.
  - Těmto intervalům lze definovat, co znamenají z hlediska kvality modelu v lidsky přijatelné podobě. Jinými slovy lze definovat doporučení uživateli, jak by mohl daný model vylepšit.

4.2 Výběr technologií pro výsledný nástroj

V této sekci představím technologie, které jsem vybral s ohledem na implementaci nástroje, jehož vizi a funkční požadavky jsem představil v předchozí sekci.

Nástroj na počítání měr kvality bude realizován jako enterprise aplikace na platformě Java EE 7. Bude se jednat o serverovou aplikaci, která bude podporovat napojení externích systémů pomocí RESTového rozhraní.

Pro ukládání informací použiji databázi PostreSQL. Na zvážení pro budoucí verze nástroje je i databáze BaseX, která je specializovaná pro ukládání XML dokumentů.

Součástí bude jednoduché a funkční grafické rozhraní. Grafické rozhraní bude realizováno pomocí technologie JSF.

4.2.1 Java Enterprise Edition

Java Enterprise Edition neboli ve zkratce Java EE[1] je edice jazyku Java od firmy Oracle pro implementaci business enterprise aplikací a webových aplikací. Java EE nabízí řadu API pro implementaci velkých a robustních aplikací. Standardní edici Java EE rozšiřuje například o Enterprise JavaBeans, Java Servlet API, Java Security API, Java Message Services a řadu dalších. Tyto i další se postarají jak o chod celé robustní aplikace, tak o bezpečnost.

Java Persistence API

Java Persistence API (zkráceně JPA[2]) je knihovna jazyku Java, jak v edici enterprise, tak v edici standard. Je využívána pro mapování datových zdrojů na objekty v jazyku Java. Synchronizace dat mezi datovým zdrojem a datovou strukturou v aplikaci je více méně zajištěna automaticky.

JavaServer Faces

JavaServer Faces (zkráceně JSF[3]) je technologie pro implementaci webového uživatelského rozhraní pro aplikace na platformě Java. JSF je technologie založená na upraveném XHTML a dovoluje pouze volání metod Managed Bean nebo CDI beans s anotací Named, které dále komunikují s business logikou. Díky tomu implementuje čistě MVC model. Tímto se například liší od technologie JSP, která dovoluje vkládat kód přímo mezi HTML a naopak.

Zároveň technologie podporuje tvorbu vlastních komponent a XHTML tagů. Díky tomu se dá efektivně využít znovu použitelnosti stejného kódu, jen s jinými parametry.

4.2.2 GlassFish Server

K běhu webové aplikace na platformě Java je zapotřebí aplikační webserver. Pro svůj nástroj zvolím server GlassFish [4] od firmy Oracle.

[1] http://www.oracle.com/technetwork/java/javaee/overview/index.html

[2] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

[3] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[4] https://glassfish.java.net

Na rozdíl od konkurenčních webserverů je vyvíjen firmou Oracle a měla by tedy být zajištěna nejlepší kompatibilita s platformou Java EE.

4.2.3 PostreSQL

PostgreSQL[5], zkráceně se též používá jen Postgres, je objektově-relační, free a open source SQL databáze. Tato databáze nabízí řadu nadstandardních funkcí, například Sequence.

Ve svém nástroji ji využiji pro ukládání obecných informací, jako jsou například informace o uživatelích nebo mírách kvality.

4.2.4 Business Process Model & Notation (BPMN)

Ve svém nástroji implementuji import procesních modelů právě ve standardu BPMN[6], jež jsem popisoval výše v sekci 2.2 o jazycích, které zaznamenávají procesní modely. Rozhodl jsem se tak kvůli jeho světové rozšířenosti. Ještě zásadnější je však jeho rozšíření v rámci výzkumu ČVUT. Notace BPMN je použita například i v Procesním portálu ČVUT. Nabízí se tak i možnost budoucího přímého napojení na Procesní portál ČVUT. Tvůrci procesních modelů v Procesním portálu ČVUT tak budou mít okamžitou odezvu o kvalitě nově vytvořeného procesního modelu.

4.2.5 Camunda

Pro práci s XML soubory ve formátu BPMN jsem zvolil volně dostupnou knihovnu Camunda[7], která obsahuje spoustu nástrojů pro práci s modely právě ve formátu BPMN včetně modelovacího nástroje.

V nástroji využiji knihovnu hlavně na parsování procesního modelu z formátu BPMN. Nabízí se i využití modelovacího nástroje v grafickém rozhraní v budoucích verzí nástroje a nabízet tak možnost přímého vytvoření a ohodnocení tvořeného procesního modelu.

Modelovací nástroj od společnosti Camunda je též použit v již zmíněném Procesním portále ČVUT, kde je tisíce hotových modelů, které se mohou být využity k výzkumným účelům.

4.2.6 Base X

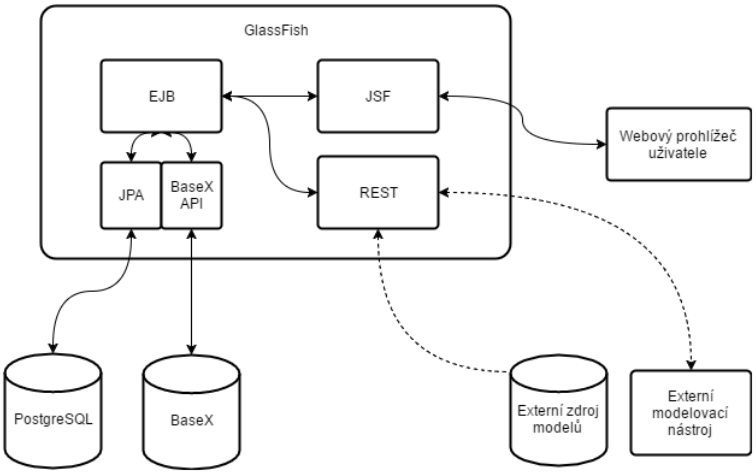
Base X[8] je databáze, která je speciálně vytvořená pro potřeby ukládání XML dokumentů. Nabízí se tedy možnost využití v budoucích verzí nástroje pro ukládání BPMN modelů, které se ukládají právě ve formátu XML.

Oproti běžné SQL databázi nabízí možnost indexace XML a vyhledávání v XML.

[5] `https://www.postgresql.org`
[6] `http://www.bpmn.org/`
[7] `https://camunda.org`
[8] `http://basex.org`

4.3 Architektura nástroje

V této sekci se zaměřuji na architektuře výsledného nástroje k dosažení vize nastíněné v předchozí sekci 4.1.



Obrázek 4.1: Architektura nástroje, zdroj: autor

Grafické rozhraní jsem se rozhodl implementovat na technologii JSF. Tato technologie umožňuje s vrstvou business logiky komunikovat přímo díky Dependenci Injection v Java EE. Což nic nemění na tom, že se jedná o naprosto oddělitelnou vrstvu. Vrstva grafického rozhraní pouze volá metody business logiky pro práci s daty. Využívá jej tedy v podstatě jen jako datový zdroj. Nic méně implementace je mnohem jednodušší, jelikož není třeba mezivrstvy RESTové služby, která zprostředkovává komunikaci mezi všemi funkcemi business logiky a grafického rozhraní. Pokud však chceme komunikovat s externími systémy, je třeba nějakého dodatečného rozhraní, např. již zmíněného RESTového rozhraní. Vezměme si klasický model architektury, kdy business logika komunikuje s grafickým rozhraním přes RESTové rozhraní. V takovém přístupu můžeme onoho rozhraní využít i pro komunikaci s externími systémy. Můj návrh toto postrádá, nic méně nevylučuje možnost RESTového rozhraní jakožto samostatné vrstvy vedle rozhraní grafického. Výhoda tohoto řešení je ta, že není třeba v tomto rozhraní implementovat veškeré funkcionality business logiky (což v případě komunikace s vrstvou grafického rozhraní by zapotřebí bylo). Stačí zajistit pouze nutnou část funkcionality pro výměnu dat. Mnou takto navržená architektura je znázorněna na diagramu 4.1.

Pro uložení informací o mírách kvality a o uživatelích navrhuji použít relační databázi PostgreSQL. Naopak se ale nehodí pro uložení XML dokumentů. Proto pro ukládání BPMN modelů navrhuji databázi BaseX. Tato databáze je speciálně určena právě pro XML dokumenty, přičemž funguje podobně jako relační databáze. Umí XML dokumenty indexovat, vybírat pouze jejich části, vyhledávat na základě určitých atributů, atd. Je to tedy mnohem vhodnější varianta pro komplexní práci s XML dokumenty.

4.3.1 Propojení s externími systémy

Připojení externích systémů se dá využít ke dvěma způsobům.

1. Datový zdroj BPMN modelů.
2. Poskytování zpětné vazby k modelům pro externí modelovací nástroje.

Hlavní myšlenkou bylo obě možnosti využít v kombinaci s Procesním portálem ČVUT, který je na ČVUT asi nejobsáhlejší databází, co se do počtu zmapovaných procesů ve standardu BPMN týče. Tedy pro výzkum měr kvality by to byl výborný zdroj. Zároveň pro editory procesů by v portále mohla být funkce zobrazování zpětné vazby v reálném čase k jejich aktuálně tvořenému procesu.

Jakožto jeden z vývojářů Procesního portálu jsem měl skvělou možnost zjistit možnosti tohoto napojení. Bohužel jsem však zjistil, že Procesní portál žádné takové napojení nepodporuje. Bude tedy potřeba dalšího vývoje i na straně Procesního portálu. Z tohoto důvodu jsem se rozhodl, že RESTové rozhraní ve svém nástroji nebudu implementovat a nechám to až na dobu, kdy se rozhodne o vývoji napojení externích aplikací i na Procesní portál.

4.4 Funkcionalita nástroje v rámci této práce

V několika předešlých sekcích jsem nastínil vizi uceleného nástroje. Zároveň jsem navrhl technologie a architekturu k naplnění této vize. V rámci rozsahu této práce však nejsem schopen zajistit veškerou výše uvedenou funkcionalitu. Rozhodl jsem se tedy, že implementuji robustní jádro s ohledem na pozdější rozšíření. Níže definuji výčet funkcionalit, které v rámci této práce implementuji.

V rámci vývoje alfa verze tohoto nástroje budu přednostně implementovat následující funkcionalitu:

- Rozlišení tří typů uživatelů, nepřihlášený uživatel, uživatel a administrátor.
- Všichni uživatelé mohou:
  - Uploadovat soubory BPMN a ZIP archivy. Nikoliv však modely ukládat.
  - Na nahraných procesních modelech počítat nástroji známé míry kvality.
  - Editovat informace o svém profilu a své přihlašovací údaje.
- Administrátor může:
  - Přidávat nové uživatele.
  - Editovat a mazat již registrované uživatele.

- Definovat nové míry kvality založené na matematických výrazech, které mohou obsahovat zbylé míry kvality jakožto matematické funkce.
- Editovat a mazat míry kvality.
- U jednotlivých měr kvality definovat zpětnou vazbu uživateli na základě výsledků dané míry, resp. na intervalu jejích hodnot.

4.5 Shrnutí kapitoly

V této kapitole jsem představil vizi výsledného nástroje pro výpočet měr kvality procesních modelů. Uvedl jsem funkční požadavky na nástroj, vybral jsem technologie pro implementaci těchto požadavků a navrhl architekturu s ohledem na tyto vybrané technologie. V neposlední řadě jsem vymezil seznam funkcionalit, které implementuji v rámci této práce.

V následující kapitole se budu věnovat už jenom funkcionalitám uvedených v sekci 4.4, avšak s ohledem na možnost budoucího rozšíření.

Kapitola 5

Implementace nástroje

V této kapitole se zabývám konkrétní implementací některých algoritmicky zajímavých částí aplikace. Tím je například reprezentace samotného modelu v rámci nástroje, implementace měr kvality a matematický evaluátor, který slouží k výpočtu komplexních[1] měr kvality, které se skládají z matematického vzorce.

5.1 Reprezentace BPMN modelu

Standard BPMN bohužel nepodporuje rozdělení komplexního procesu[2] do více souborů, kde každý podproces je ve vlastním souboru. Nic méně praxe je přesně taková, že každý podproces má vlastní soubor pro větší přehlednost. Bohužel pro míry kvality je informace o podprocesech velice důležitá. Čili je třeba si informaci o grafové struktuře podprocesů nějak uchovat. K tomuto účelu jsem navrhl vlastní strukturu, která tyto modely reprezentuje.

Camunda také nabízí API pro práci s BPMN modely. Nabízí například nalézt části modelu dle typu, jména, id a další. Nabízí dokonce i možnost přidání prvku do modelu, či odebrání prvku z modelu. Camunda přináší i možnost definice dalších vlastností k elementům díky tzv. ExtensionElement, což je element modelu, který může nést dodatečnou informaci (nad rámec BPMN standardu). Bohužel ani Camunda API nenabízí možnost reprezentace modelu ve více instancích / souborech. Rozhodl jsem se tedy využít „Extension Elementu“ k tomu, abych k elementu, který odkazuje na podproces v externím souboru, přidal informaci o jméně tohoto souboru. Do Extension Elementu je tedy přidána informace o externích souborech v podobě klíč - hodnota. Ukázka v camunda modeleru a v BPMN je v sekci 6.2.1.

Reprezentace grafového uzlu je realizována pomocí interface *IBpmnProcessNode*. Tento interface implementuje třída *BpmnProcessNode* a reprezentuje tak podproces v grafové struktuře celého procesního modelu. Je to obalující třída instance modelu (reprezentující podproces), která samotný model podprocesu rozšíří o grafové vlastnosti uzlu. Tato třída obsahuje informace o

[1] Komplexní mírou zde myslím míru, která je definovaná matematickým vzorcem, který může obsahovat další známé míry kvality.

[2] Komplexním procesem myslím proces, který se skládá z několika podprocesů a každý z nich je uložený ve vlastním externím souboru.

jméně souboru, ze kterého tento podproces pochází, v jaké hloubce grafu se tento podproces nachází, samotnou instanci BPMN modelu z Camunda API a v neposlední řadě potomky a rodiče uzlu grafu.

Celou strukturu zastřešuje třída *BpmnProcessTree*, která drží kořen grafu reprezentující model celého procesu. Není úplně přesné hovořit o kořenu, jelikož výsledný graf nemusí být strom. Kořenem zde myslím „top level“[3] proces, který je pro čtenáře procesního modelu výchozím. Též obsahuje metodu *createTreeFromHashMap()* pro sestavení této struktury z mapy[4], která obsahuje klíče jako jména souborů a jako hodnoty instance modelů, které tyto soubory obsahují. Tato metoda též obsahuje detekci cyklů. Pokud model obsahuje cykly, uloží se informace o hloubce cyklu pro pozdější použití v mírách kvality. Pokud model cykly neobsahuje, uloží se informace o maximální a průměrné hloubce grafu, též pro použití v mírách kvality.

5.2 Implementace měr kvality

Míry kvality jsou klíčovou částí aplikace. Snažil jsem se tedy udělat kvalitní objektový návrh, který bude dobře reflektovat potřeby spojené s výpočtem měr a dávání zpětné vazby uživateli o kvalitě jeho modelu. Tento návrh popisuji v následujících sekcích.

5.2.1 JPA objektová struktura

Jelikož je třeba si informace o mírách uchovávat, rozhodl jsem se, že míry zpracuji jako JPA entity. Informace o mírách tedy budou uložené v databázi a algoritmy pro výpočet daných měr budou implementovány v jednotlivých třídách reprezentující dané míry. Celou strukturu znázorňuje UML diagram 5.1.

Jak je z UML diagramu 5.1 patrné, všechny míry jsou potomci abstraktní třídy *AbstractQualityMeasure*, ze které dědí *SizeQualityMeasure*, *ComplexQualityMeasure* a další. Třída *AbstractQualityMeasure* zapouzdřuje společné atributy a metody všech měr jako je *Abbreviation*[5], *Name*, *Description* a každá míra děděná z této abstraktní třídy zpravidla jen nastavuje tyto vlastnosti a přetěžuje abstraktní metody pro výpočet míry a pro vrácení lidsky přijatelné textové zpětné vazby na procesní model.

SizeQualityMeasure je generická třída, ze které dědí všechny míry kvality typu size. Většina těchto měr kvality má stejný výpočet, jde pouze o součet prvků daného typu. Výpočet je tedy realizován v této rodičovské třídě. Třídy, které od ní dědí, zpravidla nastavují jen vlastnosti, jako je již zmíněná zkratka, jméno a popis.

ComplexQualityMeasure je třída, která reprezentuje míry vytvořené administrátorem, která obsahuje navíc atribut *Expression*, což je matematický

[3] Podproces, ze kterého čtenář vychází. Tedy podproces hloubky 0.

[4] Datová struktura typu klíč-hodnota.

[5] Abbreviation zajišťuje unikátní identifikátor míry pro použití jako matematické funkce v komplexních mírách

vzorec obsahující matematické operace a jiné míry kvality. Více se vyhodnocováním těchto měr věnuji v kapitole 5.3 Matematický evaluátor.

Všechny míry kvality jsou uloženy v databázi v jedné tabulce. O správné načtení typů se stará JPA. Typ je definován v tabulce sloupcem *measure_type*, který je u základních měr shodný se sloupcem *abbr*. Administrátorem definované komplexní míry kvality jsou rozlišovány hodnotou *Complex* ve sloupci *measure_type*.

Pro toto řešení jsem se rozhodl z důvodu, že informace o všech mírách jsou přehledně uloženy v jedné tabulce v databázi. Z pohledu budoucího rozšiřování lze snadno například vypsat výčet všech známých měr, jejich zkratek, apod.

Ke každé míře kvality se též váže záznam z tabulky *user_feedback* (JPA entity *UserFeedback*). Zde jsou uložené textové řetězce pro zpětnou vazbu uživateli založené na intervalu hodnot míry, ke které jsou vázány.

Pro úplnost uvádím seznam implementovaných měr kvality v příloze C.

5.3 Matematický evaluátor

V rámci této práce je implementován též matematický evaluátor. Slouží k vyhodnocování komplexních měr kvality, které může definovat administrátor právě za pomocí matematického vzorce v infixové notaci.

V této sekci probírám problematiku matematických notací, dále uvádím průběh vyhodnocování matematického výrazu a v neposlední řadě též popisuji algoritmus Shunting-yard, který slouží k převodu infixové notace do postfixové, a návrhový vzor Interpreter.

5.3.1 Matematické notace

Matematický vzorec můžeme zapsat několika způsoby. V předchozí části, v úvodu o matematickém evaluátoru, jsem uváděl pojmy infixová a postfixová notace. V této podsekci definuji tyto notace, ukáži, v čem se tyto notace liší a jaké mají výhody a nevýhody.

Nejčastější a lidsky nejpřívětivější forma zápisu je infixová notace. Je to notace, kdy se operátor nachází mezi operandy a funkce mají svoje parametry definované v závorkách. Příklad výrazu v infixové notaci: $(1 + 3) \cdot min(4, 2)$. Velká nevýhoda této notace je, že se velice těžko vyhodnocuje počítačem, je třeba definovat přednosti operátorů, používat závorky a další. Tyto nevýhody řeší postfixová notace.

Postfixová notace je notace, kdy operátor (funkce) je zapsán až za operandy (parametry funkce). Příklad výrazu v postfixové notaci: $1\,3\,+\,4\,2\,min\,\cdot$. Všimněte si, že se jedná o stejný výraz, pouze v jiné notaci. Výhodou je snadné vyhodnocení počítačem, kdy samotná notace řeší priority operátorů a tedy i uzávorkování. Nevýhodou pak může být, že dvě čísla jdoucí za sebou mohou splynout v jedno kvůli nedostatečné mezeře. To je však pouze nevýhoda v čitelnosti pro člověka, počítač s čísly pracuje odděleně.

Algoritmus 1 Vyhodnocení výrazu v postfixové notaci

- Procházím výraz prvek po prvku.
  - Pokud narazím na číslo, uložím jej do zásobníku.
  - Pokud narazím na operátor (funkci):
    - Vyjmu z vrcholu zásobníku n čísel dle parity operátoru (funkce).
    - Provedu operaci (vyhodnocení funkce).
    - Výsledek opět uložím na zásobník čísel.
- Při dosažení konce výrazu je na vrcholu zásobníku výsledek výrazu.

Z algoritmu 1 je patrné, že vyhodnocení postfixové notace je algoritmicky mnohem jednodušší než vyhodnocení notace infixové.

Převod infixové notace na postfixovou je v nástroji řešen pomocí Shunting-yard algoritmu popisovaný v podkapitole 5.3.4.

Mimo to existuje například i prefixová notace. Ta je velice podobná notaci postfixová jen s tím rozdílem, že operátor (funkce) je zapsána před operandy (parametry funkce). Vyhodnocení probíhá také velice podobně, jen na zásobník ukládáme operátory (funkce), dokud nenalezneme n po sobě jdoucích čísel (dle parity), abychom mohli nějaký operátor (funkci) vyhodnotit.

5.3.2 Fáze vyhodnocování výrazu

Evaluace probíhá ve čtyřech základních fázích.

1. Tokenizace vstupního matematického vzorce 5.3.3.
2. Převod do postfixové notace pomocí algoritmu Shunting-yard 5.3.4.
3. Sestavení evaluačního stromu pomocí návrhového vzoru Interpreter 5.3.5.
4. Vyhodnocení výrazu.

Celou objektovou strukturu znázorňuje UML diagram 5.2. Jednotlivé logické celky pak podrobněji probírám v následujících kapitolách 5.3.3, 5.3.4 a 5.3.5.

5.3.3 Tokenizace vstupního matematického vzorce

Třída *MathStringTokenizer* slouží k tokenizaci matematického výrazu. Neboli neformálně k nařezání výrazu na jednotlivé jeho komponenty. Tyto jednotlivé podřetězce se pak obalí do třídy *MathToken*, která obsahuje daný podřetězec, prioritu tokenu a druh tokenu. Konkrétní druhy a priority jsou popsány v tabulce 5.1.

Všechny podporované tokeny jsou definovány v třídě *SupportedMathTokens*. Zde je pole podporovaných operátorů, funkcí, levých a pravých závorek a oddělovač parametrů funkce. Všechny konkrétní podporované funkce a operátory uvádím pro úplnost v příloze D.

Druhy a priority tokenů

V tabulce níže je přehled druhů tokenů, se kterými evaluátor pracuje. Prioritu a druh nastavuje třída *MathTokenMaker*, která pracuje s konstantami definovanými ve třídě *SupportedMathTokens*. Třída *MathTokenMaker* dostane „surový“ token typu *String*, vyhodnotí, o jaký druh tokenu se jedná, a za základě toho obalí token do třídy *MathToken* s prioritou a asociativitou popsanou v tabulce. Druh tokenu je definovaný výčtovým typem *MathTokenEnum*. Asociativita je zase definovaná výčtovým typem *MathTokenAsociativityEnum*.

Druh, priorita a asociativita hrají klíčovou roli v Shunting-yard algoritmu, který popisuji v samostatné podkapitole 5.3.4.

Token	Priorita	Druh	Asociativita
Číslo	1000	*NUMBER*	-
+	2	*OPERATOR*	-
-	2	*OPERATOR*	*LEFT*
*	3	*OPERATOR*	-
/	3	*OPERATOR*	*LEFT*
^	4	*OPERATOR*	*RIGHT*
Levá závorka	0	*LEFT_BRACKET*	-
Pravá závorka	0	*RIGHT_BRACKET*	-
Funkce / Míra kvality	10	*FUNCTION*	-
Oddělovač parametrů funkce	0	*FUNCTION_PARAM_DELIMITER*	-

Tabulka 5.1: Druhy a priority tokenů matematického vzorce, zdroj: autor

5.3.4 Shunting-yard algoritmus

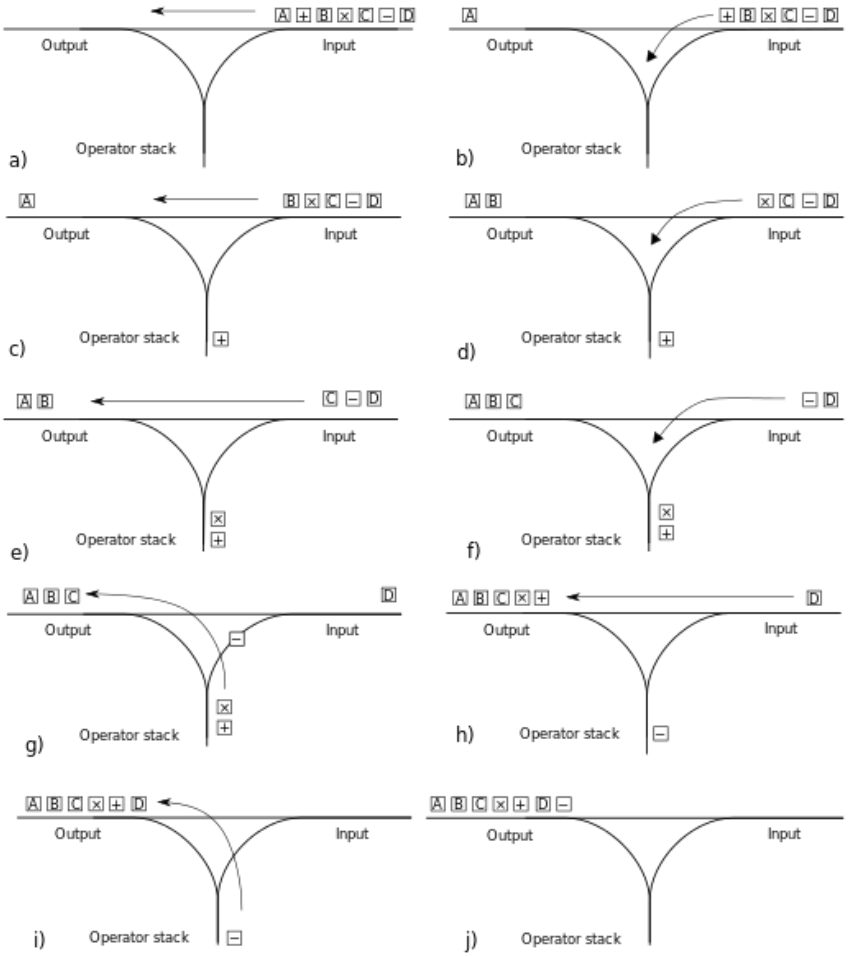
Shunting-yard algoritmus[6] (česky „algoritmus seřaďovacího nádraží“), jehož autorem je Edsger W. Dijkstra, je v nástroji využit k převodu infixové matematické notace na notaci postfixovou. Tento algoritmus popisuji zde 2.

[6] Algoritmus je popsán například na stránce `http://www.oxfordmathcenter.com/drupal7/node/628`

Algoritmus 2 Shunting-yard

- Přečti matematický token:
  - Pokud se jedná o číslo, vlož číslo na výstupní frontu.
  - Pokud se jedná o funkci, vlož jí na zásobník operátorů.
  - Pokud se jedná o operátor o_1:
    - Vlož operátor (funkci) na výstupní frontu, dokud je na vrchu zásobníku operátorů operátor (funkce) o_2, pro který platí buď:
      - Operátoru o_1 je asociativní zleva a zároveň jeho priorita je menší nebo rovna prioritě operátoru o_2. *NEBO*
      - Operátoru o_1 je asociativní zprava a zároveň jeho priorita je menší prioritě operátoru o_2.
  - Pokud se jedná o levou závorku, vlož ji na zásobník operátorů.
  - Pokud se jedná o separátor funkčních parametrů:
    - Vlož na výstup všechny operátory z vrchu zásobníku operátorů, dokud na jeho vrcholu není levá závorka (tu nevyjímat ze zásobníku).
    - Pokud na vrcholu zásobníku není dosažena levá závorka, je ve výrazu špatný počet levých a pravých závorek nebo je separátor funkčních parametrů chybně použit.
  - Pokud se jedná o pravou závorku:
    - Vlož všechny operátory z vrcholu zásobníku operátorů na výstupní frontu, dokud na vrcholu zásobníku není odpovídající levá závorky.
    - Vyjmi závorku ze zásobníku (nedávej na výstupní frontu).
    - Pokud je na vrcholu zásobníku operátorů funkční token, vlož ho na výstupní frontu.
    - Pokud se při hledání levé závorky vyprázdní zásobník, není výraz korektně uzávorkován.
- Pokud už není na vstupu žádný token:
  - Pokud zásobník operátorů není prázdný:
    - Pokud je na vrcholu zásobníku závorka, není výraz korektně uzávorkován.
    - Jinak vlož operátor na výstupní frontu.

Fungování algoritmu je názorně ukázáno na příkladu obrázku 5.3.



Obrázek 5.3: Ukázka průběhu Shunting-yard algoritmu, zdroj: [4]

5.3.5 Návrhový vzor Interpreter

Návrhový vzor Interpreter je struktura vyhodnocovacího stromu výrazu.

Tento vzor vychází z rozhraní *Expression* z výpisu 5.1, které má jednu metodu *interpret(BpmnProcessTree model)*. Jednotlivé funkce a operátory jsou pak definovány vlastní třídou, která implementuje toto rozhraní. Jako příklad uvádím *NumberExpression* ve výpisu 5.2, *PlusExpression* ve výpisu 5.3 a QualityMeasureExpression ve výpisu 5.4.

Originálně metoda *interpret()* má jako parametr mapu[7], která obsahuje proměnné a jejich dosazení. Tedy tento návrhový vzor podporuje výrazy s proměnnými a při vyhodnocování je třeba uvést dosazení za tyto proměnné. Pro účely mého nástroje je tato vlastnost návrhového vzoru nepotřebná, a proto jsem tento parametr nahradil pouze jedním parametrem a to strukturou, která reprezentuje procesní model. Je to z důvodu, že vyhodnocení výrazu je pouze pro účely výpočtu míry kvality, kde jediný potřebný parametr výrazu

[7]Datová struktura typu klíč-hodnota.

je právě BPMN model.

Jak je už z rozhraní *Expression* zřejmé, pro vyhodnocení celého výrazu, stačí na kořeni vyhodnocovacího stromu zavolat metodu *interpret()*. Kaskádně se pak zavolá vyhodnocení celého stromu.

```
public interface Expression {
    public double interpret(BpmnProcessTree
        bpmnProcessTree);
}
```

Výpis 5.1: Interface Expression, zdroj: autor

```
public class NumberExpression implements Expression{
    private final double number;

    public NumberExpression(double number) {
        this.number = number;
    }

    @Override
    public double interpret(BpmnProcessTree
        bpmnProcessTree) {
        return number;
    }
}
```

Výpis 5.2: Třída NumberExpression, zdroj: autor

```
public class PlusExpression implements Expression{
    private final Expression left;
    private final Expression right;

    public PlusExpression(Expression left, Expression
        right) {
        this.left = left;
        this.right = right;
    }

    @Override
    public double interpret(BpmnProcessTree
        bpmnProcessTree) {
        return left.interpret(bpmnProcessTree) + right.
            interpret(bpmnProcessTree);
    }
}
```

Výpis 5.3: Třída PlusExpression, zdroj: autor

```
public class QualityMeasureExpression implements
    Expression{
     private final AbstractQualityMeasure measure;

     public QualityMeasureExpression(
         AbstractQualityMeasure measure) {
          this.measure = measure;
     }

     @Override
     public double interpret(BpmnProcessTree
         bpmnProcessTree) {
          return measure.getMeasureValue(bpmnProcessTree)
              ;
     }
}
```

Výpis 5.4: Třída QualityMeasureExpression, zdroj: autor

5.3.6 Kombinace Shunting-yard algoritmu a návrhového vzoru Interpreter

Třídu *InfixExpressionEvaluator,* která vyhodnocuje matematické infixové výrazy, je kombinací algoritmu Shunting-yard a návrhového vzoru Interpreter.

Úprava Shunting-yard algoritmu je jen minimální. Místo výstupní fronty je použit výstupní zásobník na objekty typu *Expression*:

Algoritmus 3 Kombinace Shunting-yard algoritmu a návrhového vzoru Interpreter, zdroj: autor

---

- Pokud z algoritmu Shunting-yard jde na výstup číslo, vytvoř NumberExpression a vlož do zásobníku.
- Pokud z algoritmu jde na výstup operátor (funkce):
  - Vyjmi n objektů typu *Expression* (dle parity operátoru) ze zásobníku a vytvořím funkční *Expression* (například *PlusExpression*), pro operátory asociativní zleva je nutno vytvářet *Expression* objekty vyjmuté ze zásobníku v opačném pořadí. Následně tuto novou *Expression* vlož na zásobník.
  - Pokud na zásobníku není dostatek objektů, jedná se o chybné použití operátoru (funkce).
- Pokud po skončení algoritmu je na zásobníku více než jedna instance objektu *Expression*, jedná se o chybně zapsaný výraz.

Při tvorbě vyhodnocovacího stromu může vzniknout několik chyb. Ty jsou

řešeny pomocí vyhazování konkrétních Exceptions. Tyto chyby jsem již v jednotlivých podsekcích zmiňoval. Pro úplnost tyto chyby a jejich příslušné Exceptions definuji v tabulce níže.

Možná chyba	Exeption
Neznámý operátor	*UnknownOperatorException*
Špatné uzávorkování	*BracketsMismatchException*
Špatné použití operátoru (funkce)	*WrongUsageOfOperatorException*
Neznámá chyba (pravděpodobně chybný výraz)	*UnknownErrorInEvaluatingExpressionException*
Chyba v čtení vstupu	*IOException*

Tabulka 5.2: Tabulka možných chyb a Exception při tvorbě vyhodnocovacího stromu matematického infixového výrazu, zdroj: autor

5.4 Zabezpečení přihlašování

Zabezpečení přihlašování uživatelů spočívá v kontrole kombinace unikátního emailu a hesla v podobě hashe, který je uložený v databázi. Tvorba hashe je zajištěna algoritmem BCrypt. Pro tyto účely je použita knihovna jBCrypt, která jednak zajišťuje generování hashe, generování soli k heslům a kontrolu hesla.

Při registraci uživatele (případně změně hesla) je k heslu přidána tzv. sůl. Solí se myslí nějaký náhodný řetězec, který je posléze zahozen. Z hlediska bezpečnosti je výhoda ta, že dvě stejná hesla uživatelů nebudou mít stejné hashe. To je výhoda při zcizení údajů z databázi. Výhoda i nevýhoda je delší čas potřebný k ověření uživatele, jelikož chvíli trvá, než se ověří možné kombinace solí k uživatelem zadanému heslu. Výhoda spočívá v tom, že se delším časem zamezí brute force útoku. Nevýhoda je zřejmá, uživatel musí počkat o trochu déle na ověření přihlašovacích údajů než u jiných metod.

Aby se nedostal citlivý obsah do nepovolaných rukou (například administrace portálu), je v aplikaci využito filtrů. Tyto filtry jsou aplikovány na každý HTTP dotaz, který jde na URL pattern definovaný v konfiguračním souboru *web.xml*. Celé administrátorské rozhraní je mapováno na URL pattern */admin/** a přesně i na tento pattern je aplikován filter *AdminUserFilter*. Tento filtry na každý HTTP dotaz zjistí, zda je přihlášený uživatel a zda má tento uživatel administrátorská oprávnění. Pokud uživatel nevyhovuje, dojde k přesměrování na úvodní stránku.

De facto stejně funguje filter pro přístup k editaci uživatelova profilu. Tento filter je však mírnější a dovolí přístup jakémukoliv přihlášenému uživateli.

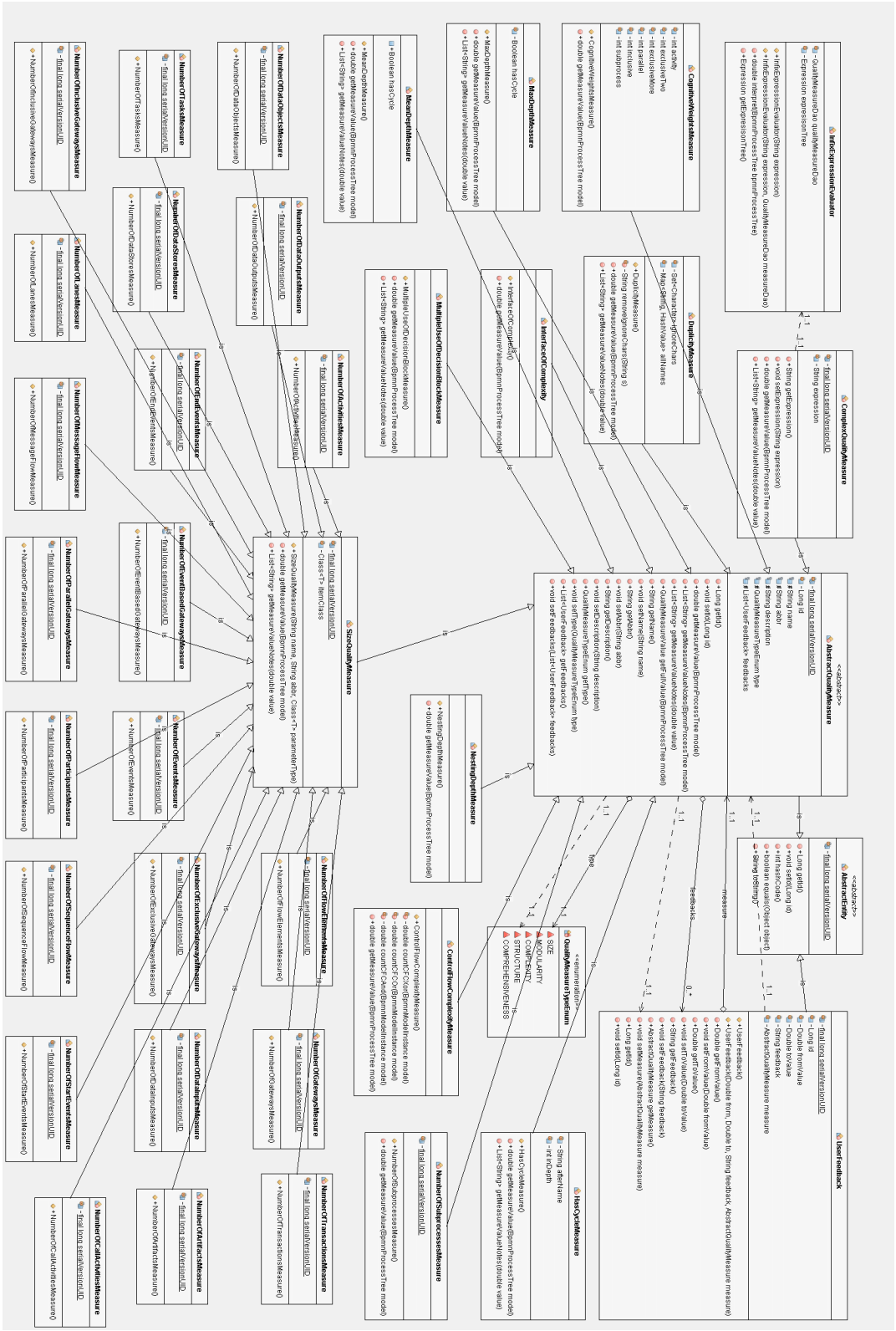
5.5 Shrnutí kapitoly

V této kapitole jsem diskutoval algoritmicky zajímavé části aplikace. Jako například reprezentaci procesního modelu v kódu, implementaci měr kvality

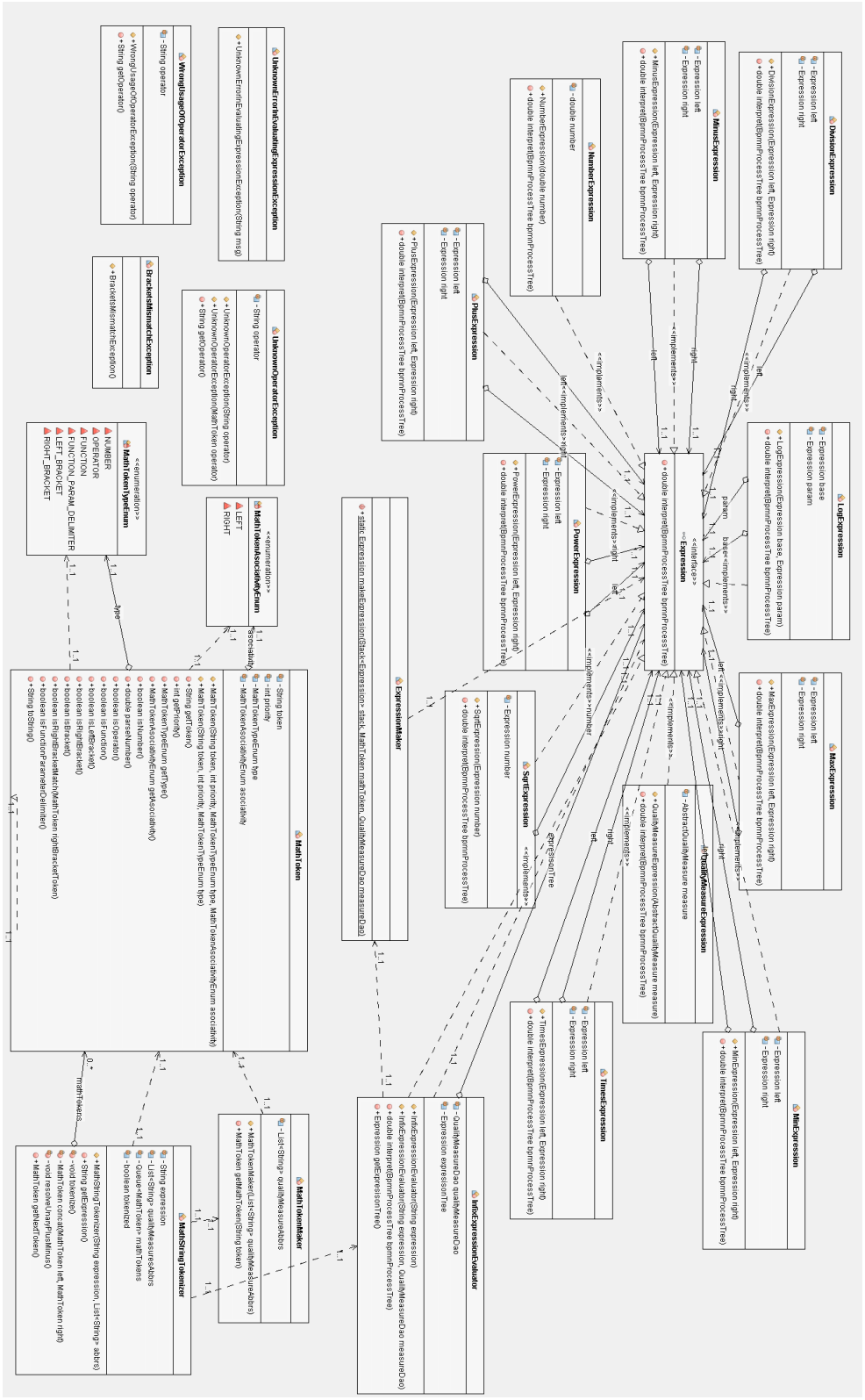
procesních modelů, vyhodnocování matematických výrazů v infixové notaci a také zabezpečení přihlašování uživatelů a citlivého obsahu.

Ostatní části aplikace jako grafické rozhraní, pomocné třídy, řídící Java beans, atd. jsou algoritmicky nezajímavé, a proto jsem se s nimi v této kapitole nezabýval. Pro více informací o implementaci je možné nahlédnout do zdrojového kódu, který je přiložen na CD. Aktuální verze zdrojového kódu je také dostupná na GIT repozitáři[8].

[8] `https://gitlab.fel.cvut.cz/zidekja2/BachelorThesis.git`



Obrázek 5.1: JPA struktura měr kvality, zdroj: autor



Obrázek 5.2: UML diagram objektové struktury matematického evaluátoru, zdroj: autor

Kapitola 6

Uživatelská příručka

V následující kapitole se zabývám používáním programu z uživatelského hlediska. V první části popisuji instalaci a nasazení aplikace, v druhé pak používání nástroje, resp. přípravu BPMN modelu pro použití v nástroji a použití jednotlivých obrazovek vytvořeného programu.

6.1 Instalace

V této sekci popisuji instalaci potřebného běhového prostředí a následné nasazení aplikace na aplikační server GlassFish.

6.1.1 Příprava běhové prostředí

Pro běh programu je potřeba prostředí Java Runtime Environment[1] (zkráceně JRE), na kterém běží nějaký aplikační server. Aplikaci jsem vyvíjel na aplikačním serveru Glassfish verze 4.1b13[2]. Pro bezproblémový chod tedy doporučuji právě aplikační server Glassfish ve verzi 4.1b13, ačkoliv by novější verze neměla dělat problémy.

Dále je třeba nainstalovat databázové prostředí PostgreSQL[3] a nastavit v aplikačním serveru GlassFish tzv. JDBC Connection, aby server věděl o připojení k databázi. Ale ještě předtím je třeba databázi připravit a vložit ovladač databáze PostgreSQL pro server GlassFish. Ten stáhneme z webu [4] a vložíme do instalačního adresáře GlassFishe, konkrétně do */glassfish/lib*. Pokud máte server spuštěný, restartujte jej.

V první řade je třeba vytvořit databázi, například jménem *bpmn_util*. Dále pak vytvoříme uživatele s možností přihlášení a s přístupovými právy k této nově vytvořené databázi. Například uživatele *bpmn_user* s heslem *bpmn-psswd*. Konkrétní postup se liší podle rozhraní, které k databázi použijete. Osobně mohu doporučit program pgAdmin[5], který je velmi intuitivní a

[1] Stažení je možné na stránce `https://www.java.com/en/download/`

[2] Stažení je možné na stránce `http://download.oracle.com/glassfish/4.1/promoted/index.html`

[3] Ke stažení na stránce `https://www.postgresql.org/download/`

[4] Ke stažení na stránce `https://jdbc.postgresql.org/download.html`

[5] Ke stažení na adrese `https://www.pgadmin.org/download/`

jednoduchý. Doporučuji spíše verzi 3. Verze 4 je nová a stále má občas nedostatky.

Za druhé je potřeba tuto databázi v GlassFishi zaregistrovat. Napřed je potřeba vytvořit tzv. Connection Pool. Nastavení nového Connection Poolu se nachází v menu „Resources->JDBC->JDBC Connection Pools“. Na první straně vytváření nového „Connection Poolu“ zadejte *libovolné jméno* (například *bpmn_util*), jako „Resource Type“ vyberte *javax.sql.ConnectionPoolDataSource*, „Database Driver Vendor“ uveďte *Postgresql* a klikněte na tlačítko „Next“.

Na druhé straně odscrollujte až naspod stránky a vyplňte následujících 5 polí: „DatabaseName“: *bpmn_util*, „PortNumber“: *5432*, „User“: *bpmn_user*, „Password“: *bpmn-psswd*, „ServerName“: *localhost*. Případně hodnoty změňte dle vašeho nastavení v předchozích krocích. Ostatní pole odstraňte.

Obě části jsou ukázány na obrázcích F.1 a F.2 v příloze F.

Poslední potřebné nastavení je JDBC Resource. Stačí pouze v menu „Resources->JDBC->JDBC Resources“ kliknout na „New“, jako „JNDI Name“ uvést *jdbc/bpmn_measure_tool*[6] a jako „Pool Name“ uvedeme námi nově vytvořený pool *bpmn_util*. Toto nastavení je znázorněno na obrázku F.3.

Další práce na databázi už nejsou třeba. Veškeré tabulky jsou automaticky vygenerovány pomocí JPA při prvním spuštění aplikace. Zároveň se třída *Starter* postará o naplnění potřebnými daty jako jsou míry kvality a defaultní administrátor.

Defaultní administrátor má email: *portal.admin@bpmn.cz* a heslo: *portal-psswd*. Samozřejmě po instalaci velice doporučuji toto heslo změnit.

6.1.2 Deploy a první spuštění

Již zkompilovaný balíček *bpmn_measure_tool.ear*, který je na přiloženém CD je třeba nahrát na aplikační server GlassFish.

Jediné, co je potřeba je otevřít stránku „Applications“ a kliknout na tlačítko „Deploy“. Na této stránce stačí vybrat balíček již zmíněný *bpmn_measure_tool.ear* a kliknout na tlačítko „Deploy“. Tato stránka je ukázána na obrázku F.4.

Po úspěšném nahrání aplikace jí můžeme spustit na URL *http://server:port/bpmn-measure-tool*. Defaultně to bývá *http://localhost:4848/bpmn-measure-tool*.

Po spuštění se můžeme přihlásit pod defaultně vygenerovaným adminem s přihlášením *portal.admin@bpmn.cz* a heslem *portal-psswd*. Hned poté doporučuji heslo, případně i email, změnit.

6.1.3 Shrnutí instalace

1. Instalace prostředí Java Runtime Environment. Ke stažení je dostupné na `https://www.java.com/en/download/`.
2. Instalace aplikačního serveru GlassFish a databáze PostgreSQL. Ke stažení jsou dostupné na `http://download.oracle.com/glassfish/4.1/promoted/index.html` a `https://www.postgresql.org/download/`.

[6] Je nutno uvést přesně v tomhle tvaru. Tato hodnota je uvedena v konfiguračním souboru *persistence.xml* ve zdrojovém kódu.

3. Stáhneme ovladač databáze PostgreSQL pro Glassfish ze stránek `https://jdbc.postgresql.org/download.html` a vložíme jej do instalační složky GlassFishe */glassfish/lib*.
4. Vytvoříme databázi a uživatele, který bude její vlastník.
5. Vytvoříme Connection Pool. Potřebné nastavení je znázorněno na obrázcích F.1 a F.2.
6. Vytvoříme JDBC Recource. Nastavení je ukázáno na obrázku F.3.
7. Provedeme Deploy balíčku *bpmn_measure_tool.ear*, jak je ukázáno na obrázku *F.4*.
8. Spustíme aplikaci na URL *http://server:port/bpmn-measure-tool*. Zpravidla *http://localhost:4848/bpmn-measure-tool*.
9. Přístup k administraci získáme přihlášením uživatele *portal.admin@bpmn.cz* s heslem *portal-psswd*.
10. Změníme heslo pro tohoto administrátora.

6.2 Používání nástroje

V této podkapitole popisuji použití nástroje z uživatelského hlediska. V první části popisuji, jak připravit procesní model pro nástroj. V následujících částech pak představuji jednotlivé obrazovky nástroje a jejich použití.

6.2.1 Příprava procesního modelu pro nástroj

Jak jsem již psal v sekci 5.1, je potřeba BPMN model rozšířit o informace, které definují vztahy mezi jednotlivými soubory. Resp. definování struktury komplexního[7] BPMN modelu, který je rozdělený do více samostatných souborů. Níže popisuji, jak takovýto model pro nástroj připravit.

Pokud má procesní model pouze jeden BPMN soubor, není třeba ho nijak dále upravovat.

Camunda modeler

Problém provázanosti souborů je řešen na platformě Camunda, tudíž je řešení v Camunda modeleru[8] velice jednoduché. Stačí kliknout na element modelu, na který chceme externí soubor navázat a v pravém panelu modeleru kliknout na kartu „Extensions“. Zde stačí kliknout na tlačítko „Add property“ a přidat dvojici Name: *external-file-name*, Value: *název_externího_soubor*. Název externího souboru může i nemusí obsahovat koncovku .bpmn.

Toto nastavení je ukázáno na obrázku F.5.

[7] Komplexním modelem je zde myšlen model, kde každý podproces je ve vlastním souboru.
[8] `https://camunda.org/download/modeler/`

Ostatní modelovací nástroje

Ačkoliv je řešení postavené na platformě Camunda, neznamená to, že není možné přidat vazbu na externí soubor i s jinými editory. Standard BPMN je pouze rozšířený standard XML. Stačí tedy v libovolném modelovacím editoru uložit model jako BPMN soubor a otevřít ho v libovolném externím textovém editoru. V tomto souboru stačí najít danou aktivitu, úkol nebo jiný element představující podproces a přidat k němu pomocí extension elementu a camunda properties daný externí soubor. Na výpisu 6.1 je ukázka přiřazení souboru *4.bpmn* k elementu Task.

```
<bpmn2:task id="Task_1r505nq" name="Nějaký task
    představující podproces">
    <bpmn2:documentation textFormat="text/x-comments" />
    <bpmn2:extensionElements>
        <camunda:properties>
            <camunda:property name="external-file-name"
                value="4" />
        </camunda:properties>
    </bpmn2:extensionElements>
</bpmn2:task>
```

Výpis 6.1: Ukázka přidání externího souboru s podprocesem ve standardu BPMN

6.2.2 Upload a míry kvality

Pokud už máte připravené všechny soubory modelu, zabalte je do zip archivu. V případě, že celý model máte v jednom souboru bpmn, není ho potřeba do archivu zabalovat.

Výsledný archiv (bpmn soubor) vyberte ve formuláři na úvodní stránce nástroje a nahrajte jej. Přesně tak, jak je znázorněno na obrázku F.6.

Po nahrání obrázku budete přesměrováni na stránku s výsledky měr kvality.

V tabulce výsledků lze snadno výsledky filtrovat. Slouží k tomu formulářová pole umístěná v horní části tabulky. Lze tak například najít pouze jednu konkrétní míru. Ukázka filtrování podle jména míry je na obrázku F.7.

Tabulku lze též řadit dle potřeby. Pro tyto účely slouží šipky umístěné v každém sloupci hlavičky tabulky.

Pro doplňující informace o výsledku nebo o míře samotné, lze kliknout na malou šipku v prvním sloupci tabulky. Dojde k rozvinutí dodatečných informací k vybranému záznamu. Ukázka je na obrázku F.7.

Vypočítané hodnoty měr kvality v nástroji zůstanou po celou dobu návštěvy aplikace nebo do doby nahrání dalšího modelu.

6.2.3 Administrace

Součástí grafického rozhraní je také administrace celého nástroje. Do administrace mají přístup pouze přihlášení uživatelé s administrátorským právem.

Správa uživatelů

Pro správu uživatelů slouží stránka „Configure users“. Zde lze přidat nového uživatele nebo editovat a mazat stávající. U uživatelů lze upravovat veškeré informace jejich profilů. Tedy křestní jméno, příjmení, email, oprávnění a heslo. O změně přihlašovacích údajů jako je email a heslo by měl mít administrátor na paměti, že by měl dát o této činnosti vědět uživateli a doporučit mu změnu hesla. Ukázka této obrazovky je na obrázku F.8.

Unikátnost emailu je vyžadována jak při přidávání uživatele nového, tak při editaci stávajícího. Je tak zajištěna unikátnost přihlašovacích údajů. Zároveň je provedena kontrola při změně oprávnění z administrátora na obyčejného uživatele. Pokud by takováto akce vedla do stavu, že by neexistoval žádný administrátor, je tato akce zamítnuta.

Správa měr kvality

Jak již několikrát bylo zmíněno, administrátor též může definovat nové míry kvality. Těm stávajícím může upravovat vlastnosti jako je jméno, zkratka, popis a zpětnou vazbu uživateli, resp. intervaly hodnot míry pro zpětnou vazbu. U administrátorem vytvořených měr lze též měnit matematický výraz, dle kterého se míra vyhodnocuje. Mazat je možné pouze administrátorem definované míry kvality. Ukázka této obrazovky je na obrázku F.9.

Při přidávání míry nové nebo editace stávající je vyžadována unikátnost její zkratky. Je to z důvodu použité této zkratky v matematickém výrazu při definici míry nové. Unikátnost je tedy nutností. Při vytváření nebo editaci komplexní míry kvality, která je definovaná matematickým výrazem, je vyžadován validní výraz (správné použití operátorů, uzávorkování, ...). Při mazání míry je zase provedena kontrola, zda nefiguruje ve vzorci jiné míry. V takovém případě je akce zamítnuta a je třeba smazat míry, na které je navázána.

Ke každé míře kvality lze definovat zpětnou vazbu uživateli, resp. interval hodnot této míry, pro který se tato zpětná vazba ukazuje. Pro využití $-\infty$ v intervalu hodnot, stačí nechat formulářové pole „From“ prázdné nebo do něj napsat „-Infinity“. To samé platí pro $+\infty$ a formulářové pole „To“, stačí pole nechat prázdné nebo do něj uvést „+Infinity“.

6.2.4 Úprava informací uživatele

Každý přihlášený uživatel může editovat jeho profil a přihlašovací údaje. Má tedy možnost měnit křestní jméno, příjmení, email a heslo. Na rozdíl od editace v administrátorské části, nemůže uživatel editovat svá oprávnění. Stejně jako v administrátorské sekci je vyžadována unikátnost emailu. Ukázka je na obrázku F.10.

6.3 Shrnutí kapitoly

V této kapitole jsem popsal, jak připravit prostředí nutné pro běh programu a také jak program spustit. Dále jsem popsal, jak připravit procesní modely pro práci s nástrojem a jak nástroj ovládat.

Všechny potřebné soubory jsou na přítomny na přiloženém CD.

Kapitola 7
Testování

V této kapitole rozebírám, jak probíhalo testování na vyvíjeném nástroji. Rozebírám pouze dvě zajímavé části a to testování měr kvality procesních modelů a testování matematického evaluátoru.

7.1 Testování měr kvality procesních modelů

Testování měr kvality procesních modelů jsem prováděl na dvou modelech stažených z Procesního portálu ČVUT. Vybíral jsem záměrně modely, které obsahovaly více podprocesů, abych ověřil i chování struktury, která model v kódu reprezentuje. Tyto modely jsem pro nástroj připravil dle uvedeného popisu z podkapitoly 6.2.1. Další nezávislé testování správné funkčnosti nástroje prováděl vedoucí práce. Jelikož výsledky byly v pořádku, účelnost testování na více modelech byla neopodstatněná a z mého pohledu považuji výše uvedený počet testovacích modelů za dostatečný.

Připravené testovací procesní modely jsem vždy do nástroje nahrál a výsledky jsem porovnával s ručně vypočítanými, očekávanými výsledky.

Dva takto připravené modely jsou k nalezení na přiloženém CD. První proces „Objednávka knih“ se skládá ze tří podprocesů, kde „top level“ proces je v souboru *diagram.bpmn*. Druhý proces „Přijímací řízení“ se také skládá ze tří podprocesů, nic méně hloubka zanoření je nižší než u předchozího, „top level“ proces je také v souboru *diagram.bpmn*.

Nabízí se možnost automatického testování pomocí Unit testů. K tomu by ale bylo zapotřebí vybrat nějaký „super testovací model“ nebo sadu modelů. Ty by zohledňovaly všechny záludnosti s počítáním měr spojenými. Tvorba takového modelu však nebyla součástí této práce.

7.2 Testování matematického evaluátoru

Pro účely testování matematického evaluátoru jsem naprogramoval sedm automatických unit testů, které testují správnou funkčnost operátorů, implementovaných matematických funkcí a jejich kombinace. Tyto testy se nachází ve třídě *MathEvaluatorTest*.

Pro každý matematický operátor jsem naprogramoval speciální test. V každém testu je vyhodnocování několik vzorců, které by měly podchycovat možné použití evaluátoru. Takové testy speciálně jsem naprogramoval pro plus (i jako unární operátor), minus (i jako unární operátor), krát, děleno, umocňování. Dále jsem naprogramoval test pro implementované matematické funkce, jako jsou například logaritmu, minimum, atd. Poslední test obsahuje vzorce pro otestování uzávorkování, kombinací operátorů a jejich priorit.

Vzorce, obsahující míry kvality, jsem testoval ručně nahráváním testovacích procesních modelů a ručním přepočítáváním vzorce, kde za funkce reprezentující míry kvality jsem dosazoval hodnoty těchto měr. Výsledek z evaluátoru, resp. konkrétní komplexní[1] míry kvality, jsem porovnával s výsledkem ručně spočítaného vzorce. Je to ze stejného důvodu, jaký popisuji výše. Pro opravdu efektivní testování by byl třeba nějaký „super testovací model“. Jeho návrh však není v rozsahu této práce.

Tyto testy jsou k nalezení ve zdrojovém kódu na přiloženém CD ve třídě *MathEvaluatorTest*.

7.3 Shrnutí kapitoly

V této kapitole jsme představil, jak probíhalo testování vznikajícího nástroje. S ohledem na rozsah a cíl této práce neuvádím veškeré detaily spojené s testováním.

Testování měr jsem prováděl na relativně malém souboru dat, nicméně nezávislé testování prováděl i vedoucí práce. Z toho vyplynul tento rozsah testování za dostatečný. Pro účely testování matematického evaluátoru a tedy i komplexních měr kvality, jsem naprogramoval sedm automatických unit testů, které jsou k nalezení ve třídě *MathEvaluatorTest* ve zdrojovém kódu na přiloženém CD. Aktuální verze zdrojového kódu je také dostupná na GIT repozitáři[2].

Zároveň jsem diskutoval možné automatizace a zlepšení testování v budoucích verzích nástroje.

[1] Komplexní mírou je zde myšlena míra určená matematickým vzorcem, který je složeninou z ostatních měr kvality.

[2] `https://gitlab.fel.cvut.cz/zidekja2/BachelorThesis.git`

Část III

Závěr a přílohy

Kapitola 8

Závěr

V rámci této práce vznikl nástroj na podporu výzkumu měr kvality procesních modelů. Ten doposud probíhal převážně ručním počítáním a empirickým výzkumem. Podařilo se tedy přenést část úsilí do nástroje a některé části výzkumu automatizovat.

Nástroj umí nahrávat procesní modely v notaci BPMN a to i procesy, které mají i několik podprocesů umístěných v externích souborech. Tyto modely mohou být přímo stažitelné z Procesního portálu ČVUT, který tak nabízí velkou databázi vstupních dat pro tento nástroj. Nad těmito nahranými modely umí počítat nástroji známé míry kvality. V těchto výsledcích se dá filtrovat, řadit je a uživatel získá zpětnou vazbu nejenom z číselné hodnoty dané míry, ale i v lidsky přijatelnější, textové podobě.

Administrátor může definovat i míry nové, jakožto matematický vzorec, ve kterém může uvést již nástroji známé míry jako matematickou funkci. Zároveň může registrovat nové uživatele ať už s administrátorským právem nebo pouze jako obyčejného uživatele. Ty si zatím mohou pouze upravovat informace o svém profilu. Ale z pohledu budoucího vývoje nástroje je možnost registrace obyčejného uživatele přínosem.

Nutno podotknout, že k dosažení vize představené v sekci 4.1 je potřeba ještě dalšího vývoje. Při návrhu architektury i při implementaci jsem však tento fakt zohledňoval a vývoj dalších funkcionalit nad rámec této práce nebude problém. Zároveň jsem se při návrhu architektury rozhodl pozměnit počáteční požadavek na propojení prezenční vrstvy aplikace s vrstvou business logiky přes RESTové rozhraní. Místo toho jsem využil vlastnosti Dependency Injection technologie JSF, které propojení s business logikou výrazně ulehčuje. Zároveň to ale nevylučuje budoucího vývoje RESTové služby jakožto sekundárního rozhraní vedle rozhraní grafického.

V neposlední řadě bych rád zmínil, že výsledky mé práce budeme s vedoucím Radkem Hronzou prezentovat na konferenci Albína Bráfa dne 31. května 2017 na Masarykově ústavu ČVUT v Praze.

Práce na nástroji mě bavila a byla pro mě přínosem, jak z hlediska teoretického o procesním řízení, resp. o procesních modelech, tak z hlediska technického a implementačního, když jsem řešil některé algoritmicky zajímavé části.

Přílohy

Příloha A

Literatura

[1] Radek Hronza, Josef Pavlíček, and Pavel Náplava. Míry kvality procesních modelů vytvořených v notaci bpmn. URL: http://aip.vse.cz/index.php/aip/article/view/113, doi:10.18267/j.aip.66.

[2] Richard Mach. Návrh a tvorba nástroje pro optimalizaci procesu na základe analýzy bpm modelu, 2015.

[3] Klára Jelínková. Návrh měr kvality obchodních procesních modelu, 2017.

[4] Shunting-yard algorithm, 2001. URL: https://en.wikipedia.org/wiki/Shunting-yard_algorithm.

[5] Kiran Garimella, Michael Lees, , and Bruce Williams. *BPM basics for dummies*. Wiley, Hoboken, NJ, software ag special edition. edition, 2008.

[6] Mathias Weske. *Business process management*. Springer, Berlin, 1 edition, c2007.

[7] Václav Řepa. *Procesně řízená organizace*. Grada, Praha, 1. vyd. edition, 2012.

[8] W. M. P. VAN DER AALST. The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers*, vol. 08(issue 01):21–66, 1998. URL: http://www.worldscientific.com/doi/abs/10.1142/S0218126698000043, doi:10.1142/S0218126698000043.

[9] John Krogstie, Guttorm Sindre, and Håvard Jørgensen. Process models representing knowledge for action. *European Journal of Information Systems*, vol. 15(issue 1):91–102, 2006. URL: http://link.springer.com/10.1057/palgrave.ejis.3000598, doi:10.1057/palgrave.ejis.3000598.

[10] O.I. Lindland, G. Sindre, and A. Solvberg. Understanding quality in conceptual modeling. *IEEE Software*, vol. 11(issue 2):42–49, 1994. URL: http://ieeexplore.ieee.org/document/268955/, doi:10.1109/52.268955.

[11] Reinhard Schuette and Thomas Rotthowe. The guidelines of modeling – an approach to enhance the quality in information models. page 240. URL: http://link.springer.com/10.1007/978-3-540-49524-6_20, doi:10.1007/978-3-540-49524-6_20.

[12] H. James Nelson, Geert Poels, Marcela Genero, and Mario Piattini. A conceptual modeling quality framework. *Software Quality Journal*, vol. 20(issue 1):201–228, 2012. URL: http://link.springer.com/10.1007/s11219-011-9136-9, doi:10.1007/s11219-011-9136-9.

[13] J. Mendling, H.A. Reijers, and W.M.P. van der Aalst. Seven process modeling guidelines (7pmg). *Information and Software Technology*, vol. 52(issue 2):127–136, 2010. URL: http://linkinghub.elsevier.com/retrieve/pii/S0950584909001268, doi:10.1016/j.infsof.2009.08.004.

[14] Radek Hronza, Josef Pavlíček, Richard Mach, and Pavel Náplava. Míry kvality v procesním modelování. *Acta Informatica Pragensia*, vol. 4(issue 1):18–29, 2015. URL: http://aip.vse.cz/index.php/aip/article/view/93, doi:10.18267/j.aip.57.

[15] Uml: Unified modeling language, 2008. URL: http://www.uml.org/.

[16] Omg: Object management group. URL: http://www.omg.org/.

[17] Xpdl: Xml process definition language, 2012. URL: http://www.xpdl.org/.

[18] Wfmc: Workflow management coalition. URL: http://www.wfmc.org/.

[19] Bpmn: Business process model & notation, 2014. URL: http://www.bpmn.org/.

[20] August-Wilhelm Scheer, Oliver Thomas, and Otmar Adam. Process modeling using event-driven process chains. *Process-Aware Information Systems*, page 119, 2005-09-02. URL: http://doi.wiley.com/10.1002/0471741442.ch6, doi:10.1002/0471741442.ch6.

[21] David R. Wright. Finite state machines. 2005. URL: http://www4.ncsu.edu/~drwrigh3/docs/courses/csc216/fsm-notes.pdf.

[22] Albert. Fleischmann. *Subject-oriented business process management.* Springer, New York, 2012.

[23] Arthur Ter Hofstede. *Modern business process automation.* Springer, Heidelberg, c2010.

[24] Radek Hronza. Procesní portál – od motivace až po současnost, 2015. URL: http://blog.czm-cvut.cz/procesni-portal.

[25] Marek Neumann. Míry kvality procesních modelů, 2016. URL: https://dspace.cvut.cz/handle/10467/66210.

[26] Martina Lassaková. Návrh a tvorba měr pro výpočet kvality procesních modelů, 2016. URL: https://dspace.cvut.cz/handle/10467/66198.

[27] Josef Pavlicek, Radek Hronza, and Petra Pavlickova. Educational business process model skills improvement. page 172. URL: http://link.springer.com/10.1007/978-3-319-49454-8_12, doi:10.1007/978-3-319-49454-8_12.

[28] Radek Hronza, Josef Pavlíček, Marek Neumann, and Martina Lassaková. Řízení kvality procesních diagramů vytvořených v notaci bpmn. pages s57–62, 2016.

[29] Volker Gruhn and Ralf Laue. Approaches for business process model complexity metrics. *Technologies for Business Information Systems*, page 13, 2007. URL: http://link.springer.com/10.1007/1-4020-5634-6_2, doi:10.1007/1-4020-5634-6_2.

Příloha B

Seznam zkratek

7PMG	Seven Process Modeling Guidelines
API	Application Programming Interface
BPM	Business Process Management
BPMN	Business Process Model & Notation
CFC	Control Flow Complexity
CNC	Coeficient of Network Complexity
CW	Cognitive weight
EPC	Event-driven Process Chain
FSM	Finite State Machine
GoM	The Guidelines of Modeling
HC	Has Cycle
IoC	Interface of Complexity
Java EE	Java Enterprise Edition
JPA	Java Persistence API
JSF	Java Server Faces
ND	Nesting Depth
NOA	Number of Activities
NOD	Number of Data
NOE	Number of Events
NOFE	Number of Flow Elements
NOG	Number of Gateways

REST	Representational State Transfer
S-BPM	Subject Oriented Business Process Management
UML	Unified Modeling Language
XPDL	XML Process Definition Language
YAWL	Yet Another Workflow Language
ČVUT	České vysoké učení technické

Příloha C

Implementované míry kvality

Zde uvádím seznam implementovaných měr. V závorkách uvádím jejich defaultně nastavené zkratky (abbreviation).

- Complexity
  - Control Flow Complexity (CFC)
  - Has Cycle (HC)
- Comprehensiveness
  - Cognitive Weights (CW)
- Modularity
  - Max Depth (MaxD)
  - Mean Depth (MeanD)
- Size
  - Number of Activities (NOA)
  - Number of Artifacts (NOAtr)
  - Number of Call Activities (NOCA)
  - Number of Data Inputs (NODI)
  - Number of Data (NOD)
  - Number of Data Outputs (NODO)
  - Number of Data Stores (NODS)
  - Number of End Events (NOEE)
  - Number of Event Based Gateways (NOEBG)
  - Number of Events (NOE)
  - Number of Exclusive Gateways (NOEG)
  - Number of Flow Elements (NOFE)
  - Number of Gateways (NOG)
  - Number of Inclusive Gateways (NOIG)

- Number of Lanes (NOL)
- Number of Message Flows (NOMF)
- Number of Parallel Gateways (NOPG)
- Number of Participants (NOP)
- Number of Sequence Flows (NOSF)
- Number of Start Events (NOSE)
- Number of Subprocesses (NOS)
- Number of Tasks (NOT)
- Number of Transactions (NOTr)

- Structure
  - Number of Duplicities (NODup)
  - Interface of Complexity (IoC)
  - Multiple Use Of Decision Block In Direct Response (MUODB)
  - Nesting Depth (ND)

Příloha D

Podporované matematické operace evaluátoru

V této příloze uvádím seznam všech podporovaných matematických operátorů a funkcí v implementovaném evaluátoru. Písmena jsou proměnné, které zastupují další matematický výraz. Pokud se liší běžně psaná forma výrazu od výrazu, který je napsaný na klávesnici, je tato odlišnost uvedena v závorce.

- Operátory
  - $a + b$
  - $+a$
  - $a - b$
  - $-b$
  - $a \cdot b$ (a * b)
  - $a : b$ (a / b)
  - a^b (a ^ b)
- Funkce
  - $min(a, b)$
  - $max(a, b)$
  - $sqrt(a) = \sqrt{a}$
  - $log(a, b) = log_a(b)$
- Míry kvality
  - Jakákoliv nástroji známá míra kvality ve tvaru $Abbreviation()$
- Závorky
  - ( )
  - [ ]
  - { }

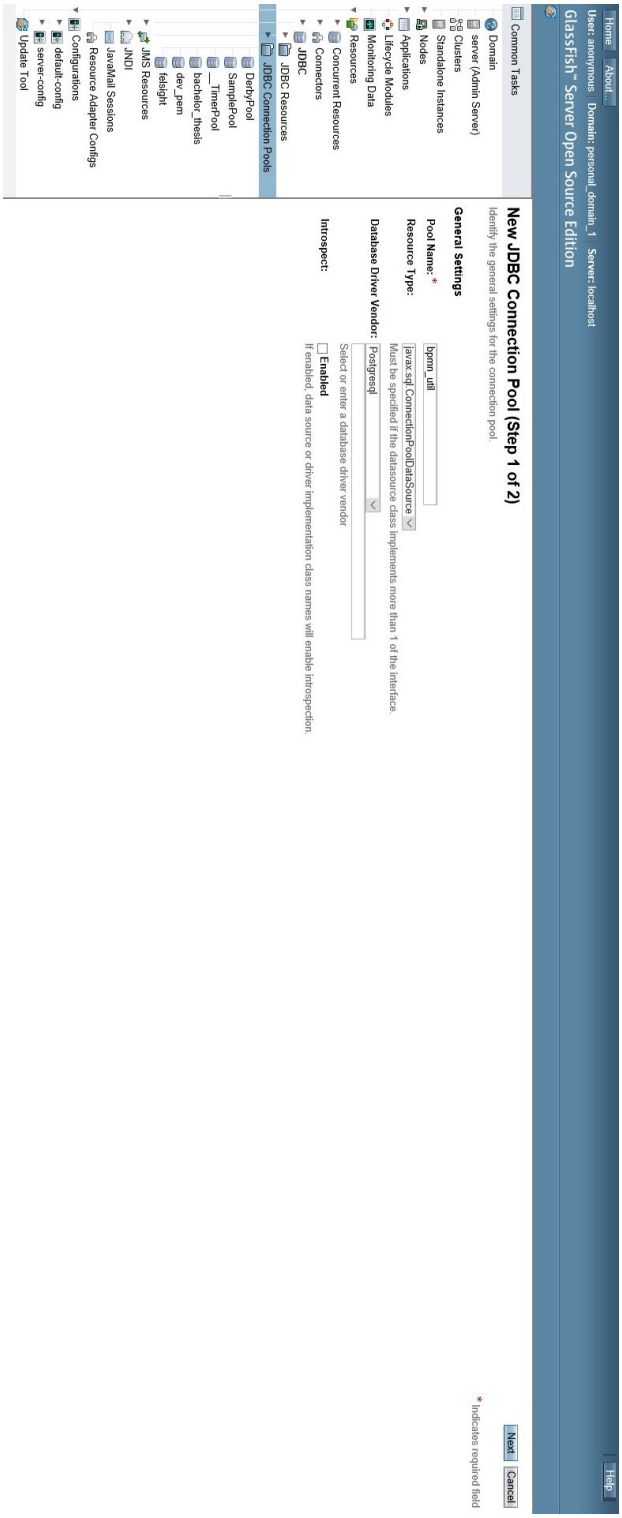
Příloha E

Seznam použitých technologií a nástrojů

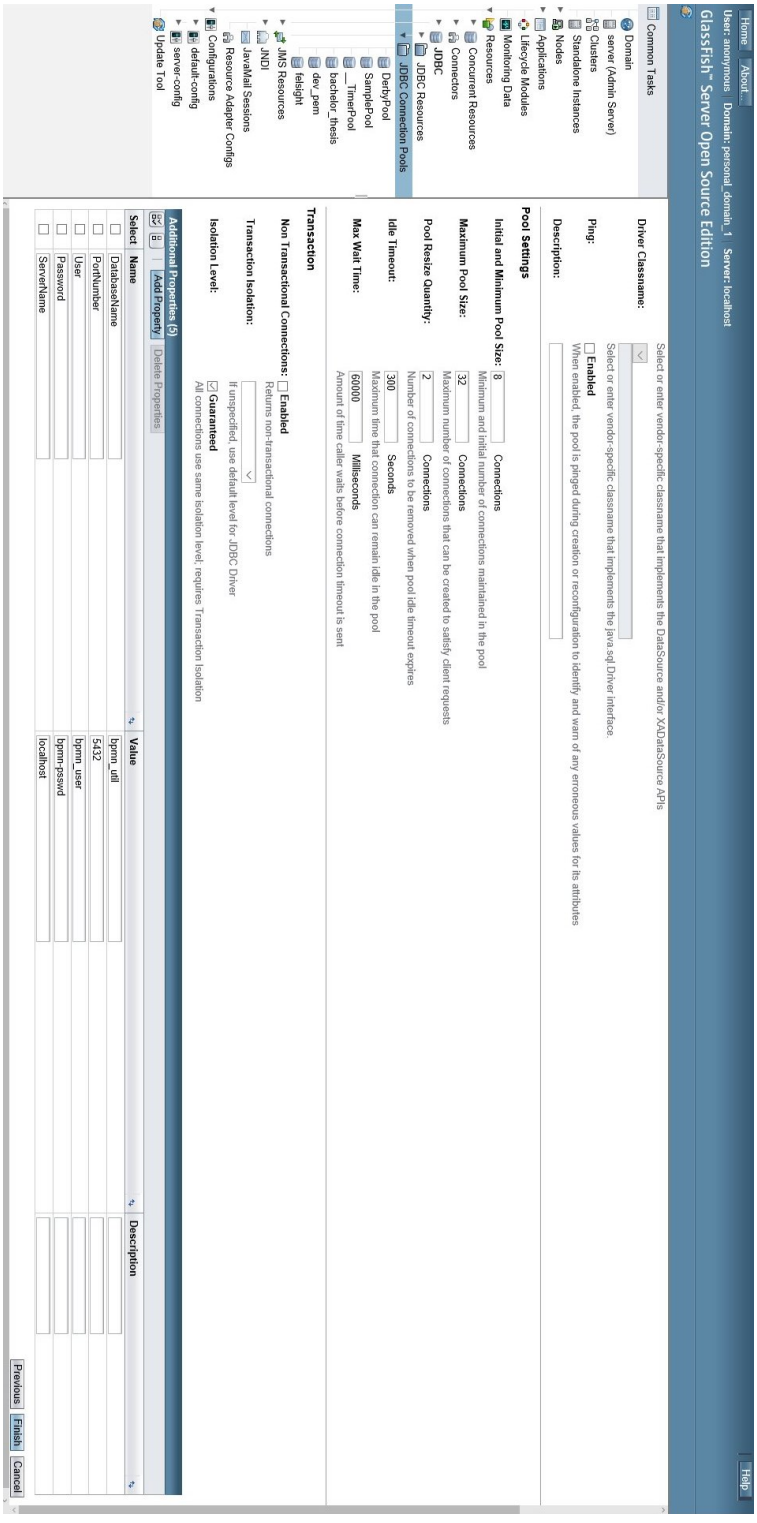
- Technologie
  - GlassFish 4.1b13
  - PostgreSQL 9.6
  - Java EE 7
    - Java Server Faces 2.2.14
      - PrimeFaces 5.3
    - jBcrypt 0.3
  - Camunda BPMN Model Api 7.6
  - HTML 5
  - CSS 3
    - Bootstrap 3.3.7
  - jQuery 3.2.1
- Nástroje
  - pgAdmin III
  - Camunda Modeler 1.6.0
  - NetBeans 8.2

Příloha F

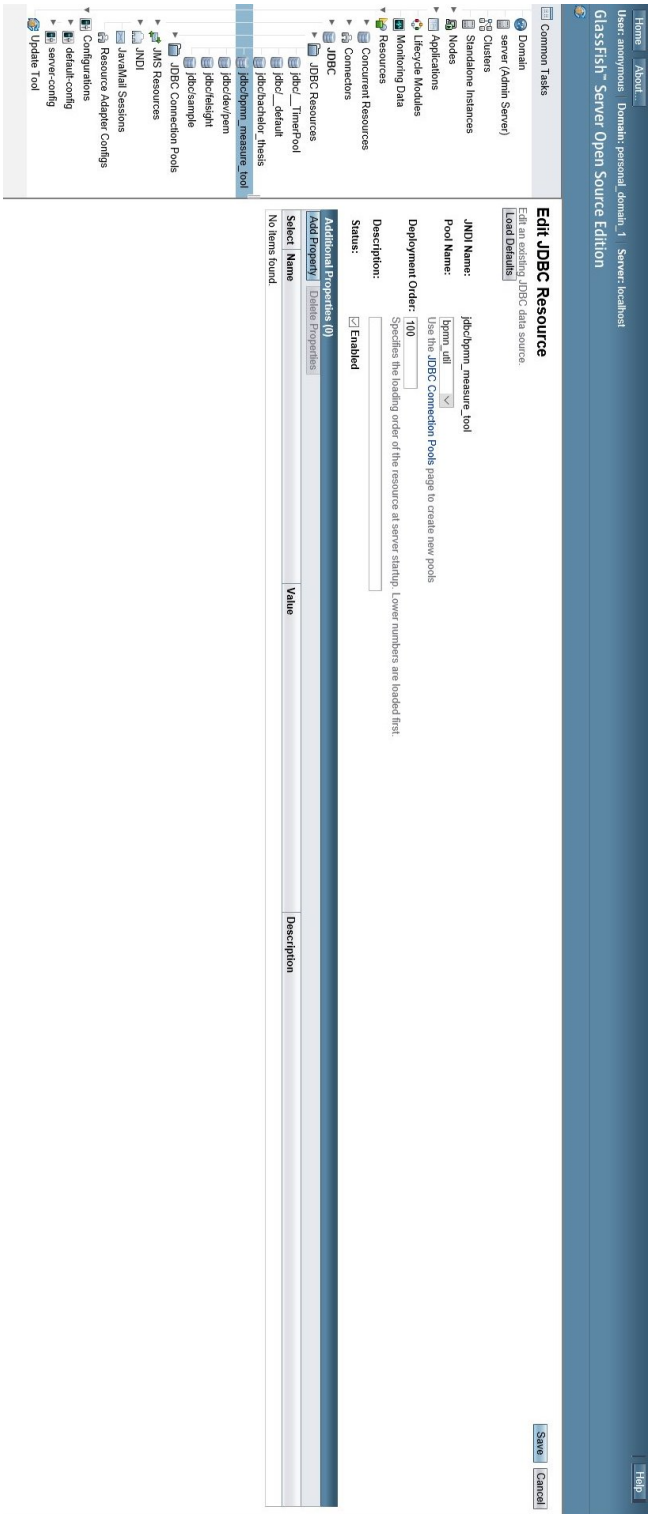
Obrazovky z instalace a běhu programu



Obrázek F.1: Tvorba Connection Pool 1/2, zdroj: autor

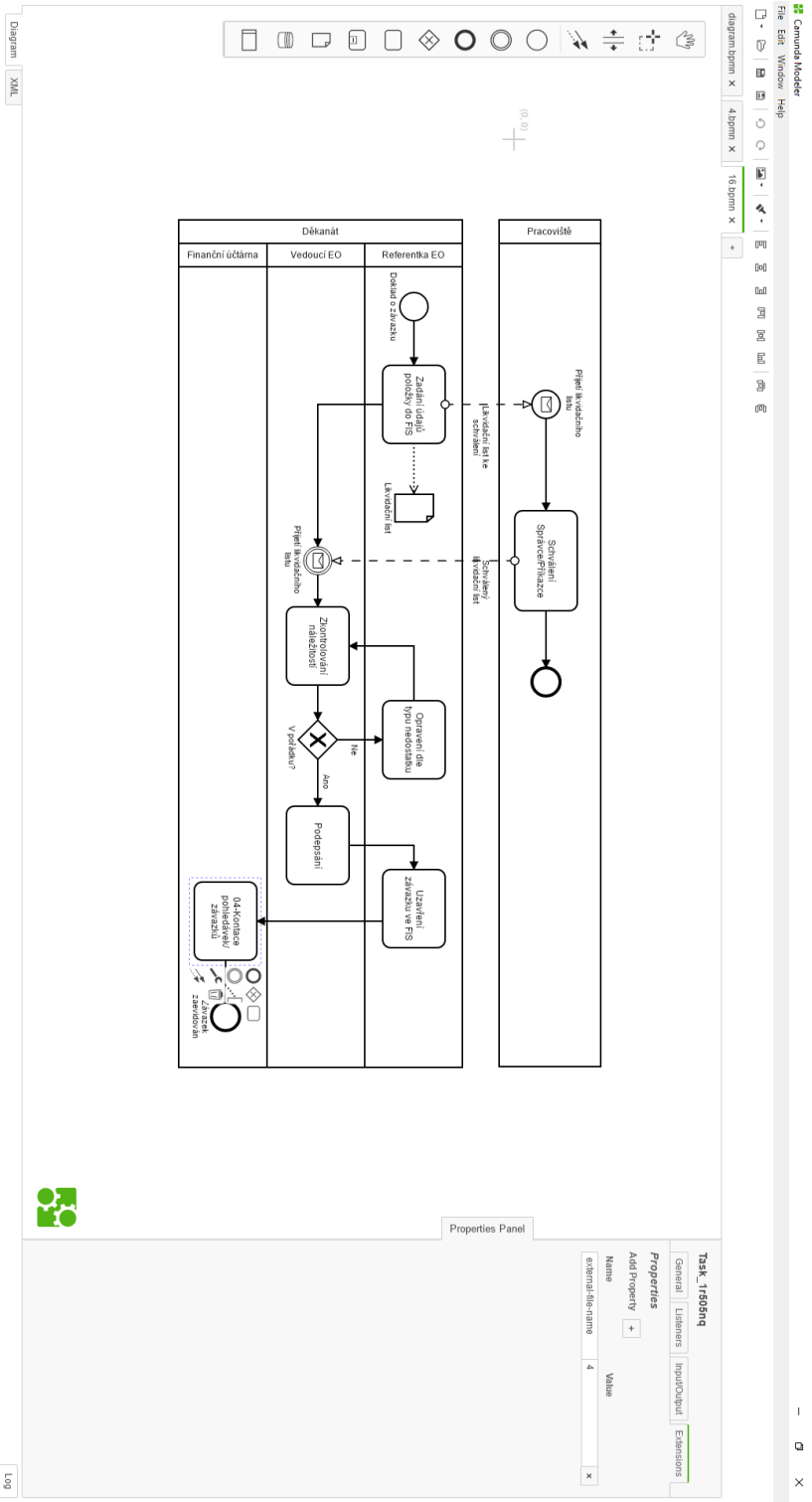


Obrázek F.2: Tvorba Connection Pool 2/2, zdroj: autor

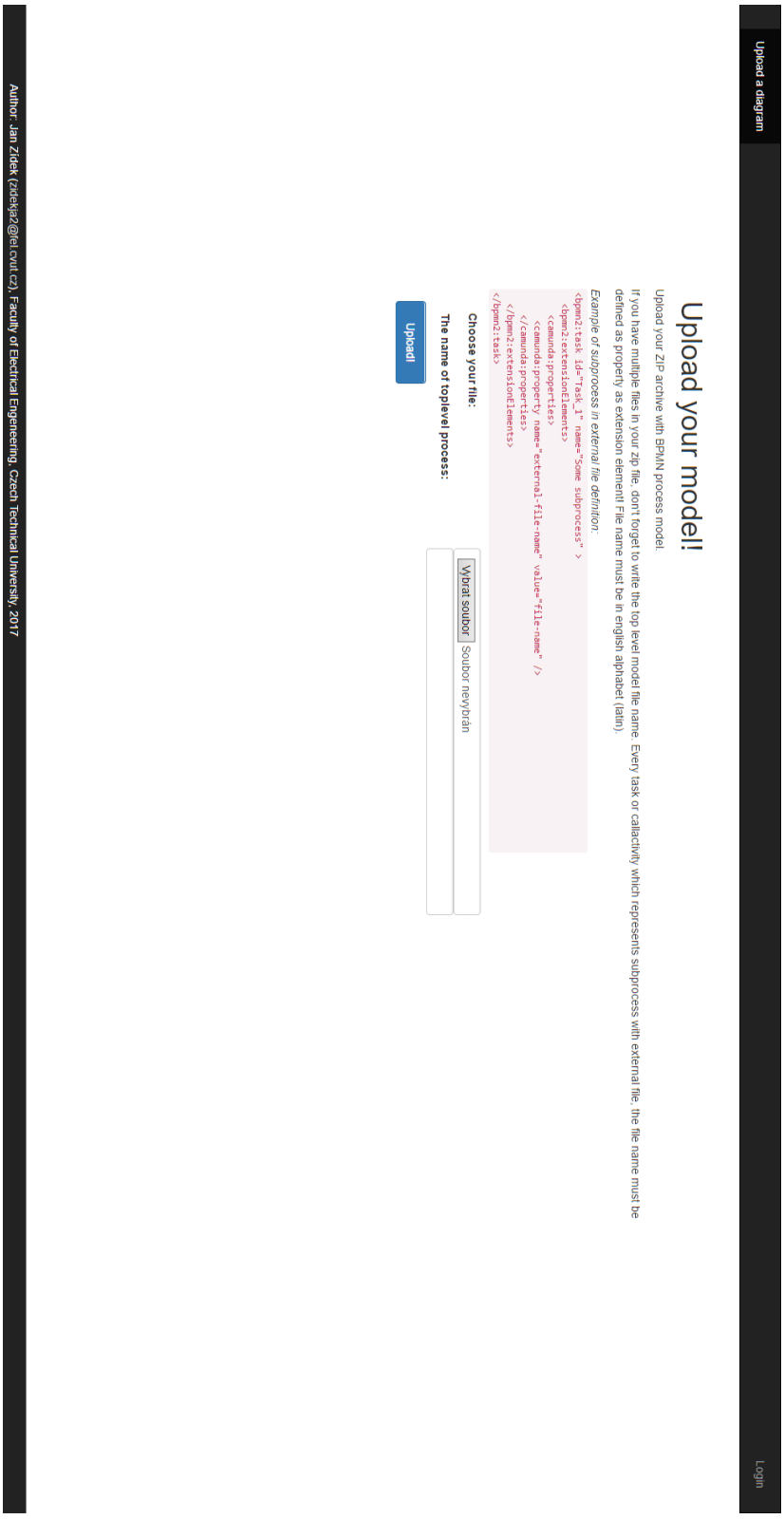


Obrázek F.3: JDBC Resource, zdroj: autor

Obrázek F.4: Deploy aplikace, zdroj: autor

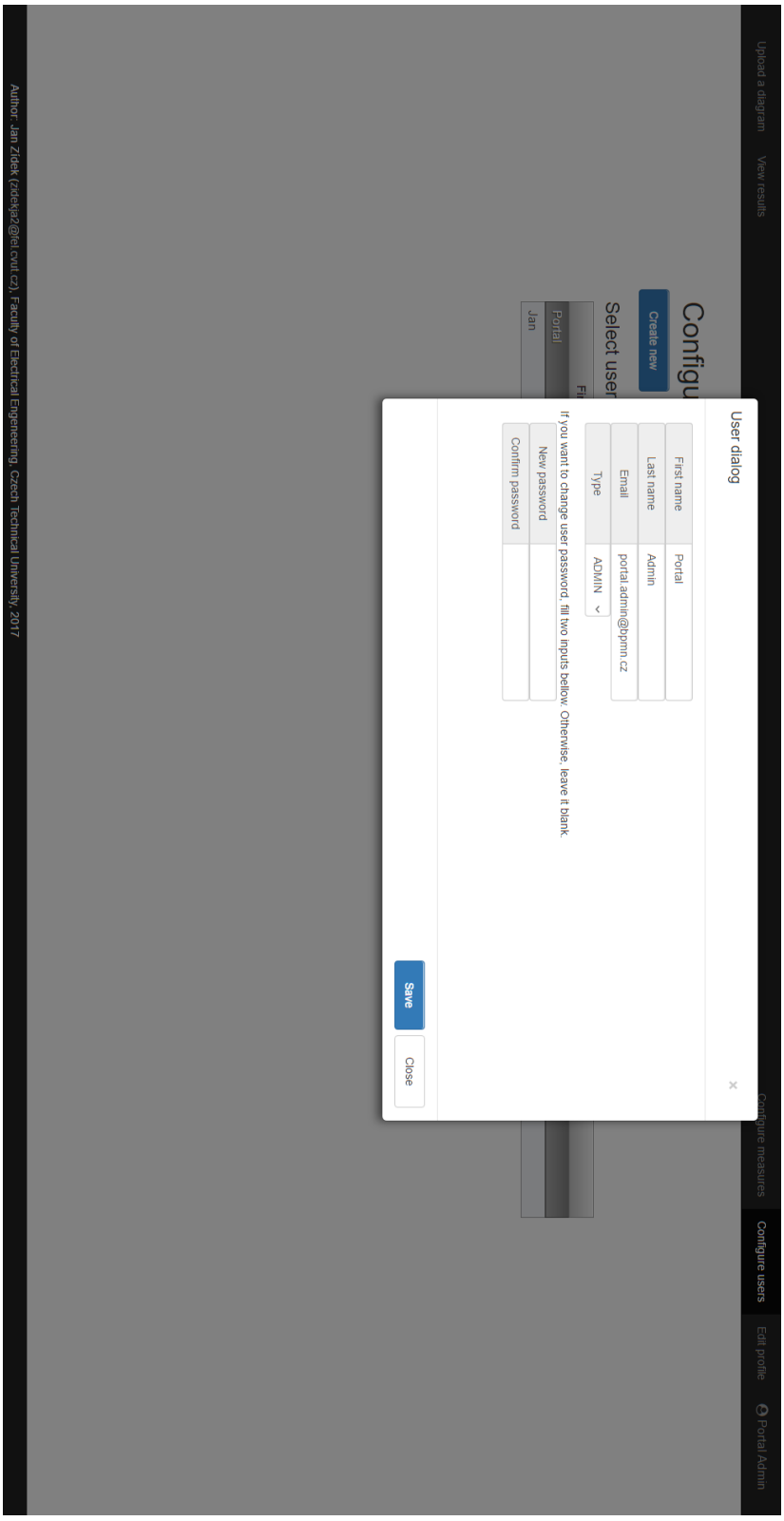


Obrázek F.5: Ukázka nastavení externího souboru podprocesu umístěného v souboru 4.bpmn v Camunda modeleru, zdroj: autor

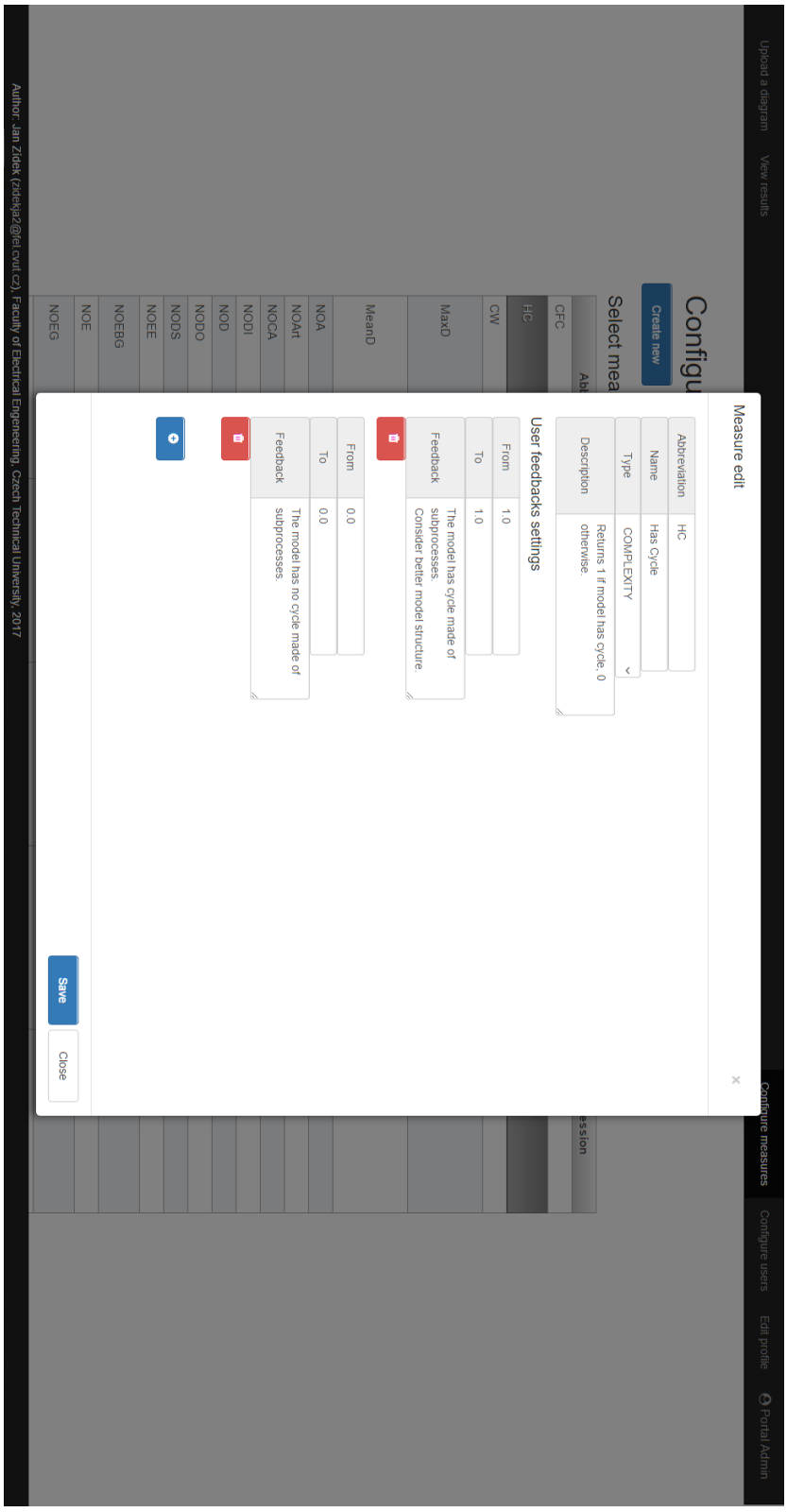


Obrázek F.6: Obrazovka formuláře pro nahrání modelu, zdroj: autor

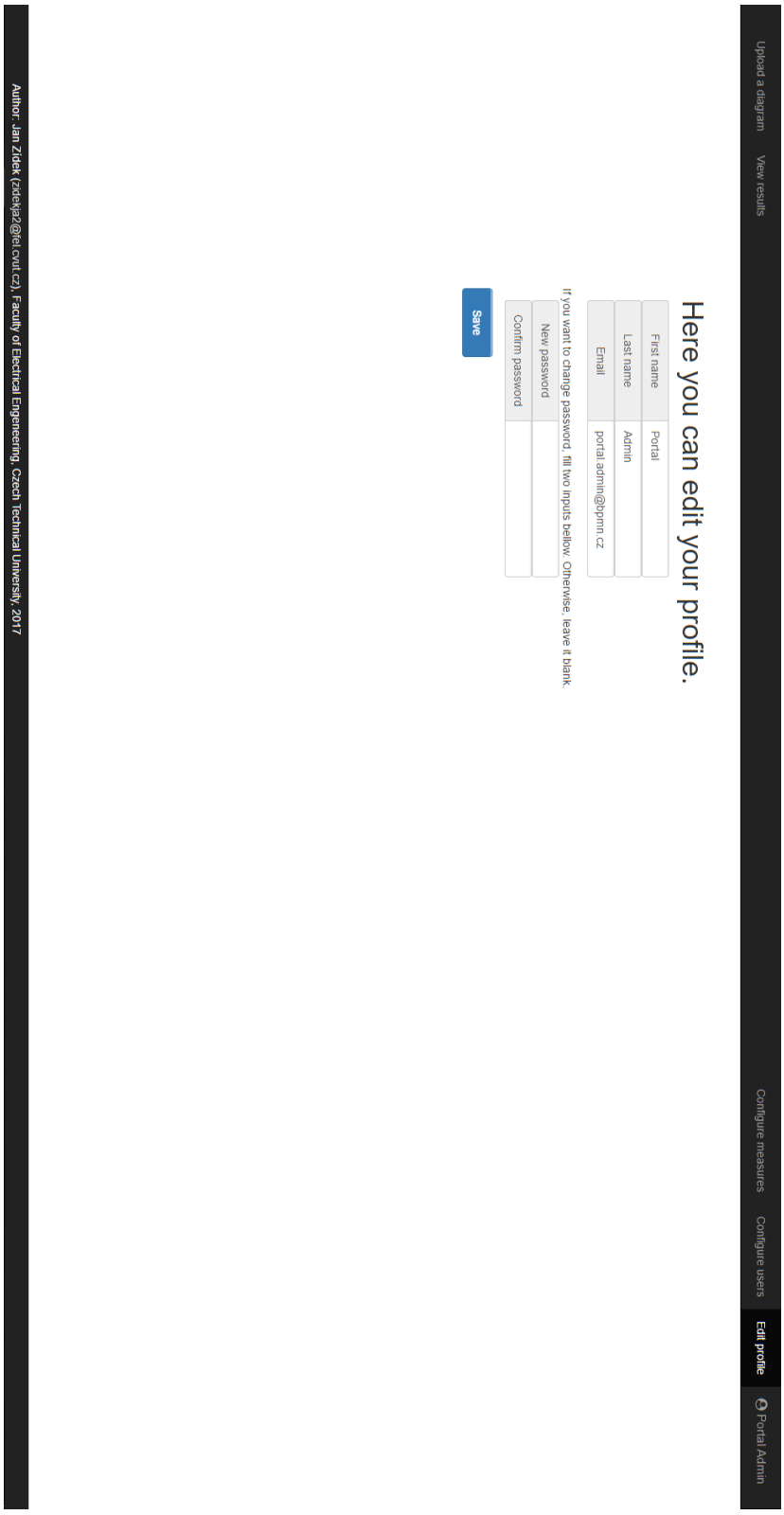
Obrázek F.7: Obrazovka přehledu výsledků měr kvality s aktivním filtrováním, zdroj: autor



Obrázek F.8: Ukázka správy uživatelů, zdroj: autor



Obrázek F.9: Ukázka správy měr kvality, zdroj: autor



Obrázek F.10: Obrazovka změny profilu přihlášeného uživatele, zdroj: autor