ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Stoklasová** Jméno: **Jitka** Osobní číslo: **457781**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vizualizace použití artefaktů procesní aplikace

Název bakalářské práce anglicky:

Visualization of process application artifacts usage

Pokyny pro vypracování:

Analyzujte systém verzování procesních aplikací na platformě IBM BPM. Navrhněte a implementujte nástroj pro analýzu artefaktů, která uživateli zobrazí textovou i grafickou informaci, v jaké části aplikace je daný artefakt použitý. Funkčnost nástroje ověřte pomocí definovaných uživatelských scénářů a testů. Vytvořte uživatelskou příručku.

Seznam doporučené literatury:

[1] GRASSEOVÁ, Monika, Radek DUBEC a Roman HORÁK. Procesní řízení ve veřejném sektoru: teoretická východiska a praktické příklady. Vyd. 1. Brno: Computer Press, 2008. v, 266. ISBN 9788025119877.
[2] SVOZILOVÁ, Alena. Zlepšování podnikových procesů. 1. vyd. Praha: Grada, 2011. Expert. ISBN 978-80-247-3938-0.
[3] SILVER, Bruce (Bruce Richard). BPMN method and style : with BPMN implementer's guide. Aptos, Calif. : Cody-Cassidy Press, 2011. ISBN 9780982368114
[4] KOLBAN, Neil. Kolban's Book on IBM Business Process Management [Online]. December 2014, Dostupné z: http://neilkolban.com/ibm/wp-content/uploads/2014/12/Kolbans-IBPM-Book-2014-12.pdf

Jméno a pracoviště vedoucí(ho) bakalářské práce:

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

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

Datum zadání bakalářské práce: **21.01.2019** Termín odevzdání bakalářské práce: **07.01.2020**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

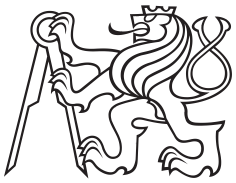
III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studentky

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Vizualizace použití artefaktů procesní aplikace

Jitka Stoklasová

Vedoucí: Ing. Pavel Náplava, Ph.D.
Obor: Software
Prosinec 2019

Poděkování

Děkuji za rady, připomínky a čas mi věnovaný při psaní mojí bakalářské práce Ing. Pavlu Náplavovi, Ph.D. a Bc. Tomášovi Malinkovičovi. Dále bych chtěla poděkovat Bc. Anně Márii Hriadelové, Bc. Tomášovi Malinkovičovi, Bc. Davidu Löfflerovi a Petru Švecovi za otestování aplikace.

Prohlášení

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

V Praze, 14. prosince 2019

Abstrakt

Bakalářská práce se zabývá analýzou systému verzujícího procesní aplikaci na platformě IBM BPM. Návrhem a implementací aplikace pro vyhledávání specifikovaných artefaktů podle názvu. Aplikace zobrazí uživateli textovou a grafickou informaci, v jaké části aplikace je daný artefakt použitý.

Klíčová slova: artefakt, vyhledávání, XML, IBM BPM

Vedoucí: Ing. Pavel Náplava, Ph.D.

Abstract

The bachelor thesis deals with analyzing of the system versioning process application on IBM BPM platform. Designing and implementing an application for searching of artifacts specified by name. The application displays the text and graphic information to the user in which part of the process application that artifact is used.

Keywords: artifact, search, XML, IBM BPM

Title translation: Visualization of process application artifacts usage

Obsah

Obrázky

Kapitola 1
Úvod

Cílem práce je analyzovat systém verzování procesních aplikací na platformě IBM BPM. Navrhnout a implementovat nástroj pro analýzu artefaktů, který uživateli zobrazí textovou i grafickou informaci o tom, v jaké části aplikace je daný artefakt použitý. Na závěr vytvoření uživatelské příručky.

Na základě úvodní konzultace s vedoucím práce panem Náplavou a odborným konzultantem panem Malinkovičem vyplynulo, že procesní designer na platformě IBM BPM nemá dostatečnou funkcionalitu ve vyhledávání použití artefaktů. Ne vždy je artefakt při použití v jiném diagramu pojmenován stejně, jako je původní jméno artefaktu. Je dobré mít tedy nástroj, který nalezne použití artefaktu, abychom nemuseli složitě procházet veškeré implementace artefaktů všech diagramů v procesní aplikaci. Čím větší procesní aplikace je, tím hůře se v ní vyhledává použití artefaktů ručně.

Do teoretické části sepíšu analýzu procesního designeru na platformě IBM BPM. Dále pak napíšu analýzu metod na parsování XML souborů a na vizualizaci nalezených diagramů.

Poté se přesunu k návrhu implementace, kde popíšu funkční a nefunkční požadavky aplikace, nakreslím diagram případů užití a diagram aktivit. Dále vytvořím UML diagram tříd a popíšu účel tříd.

V implementační části popíšu implementaci parsování XML souborů, jenž jsou interpretací procesního artefaktu. Dále pak popíšu vyhledávání artefaktů v nich a vizualizaci diagramů, ve kterých je použit vyhledávaný artefakt.

Práci zakončím otestováním aplikace pomocí unit testů, integračních testů a testovacích scénářů. Na které navážu poslední kapitolu, kde sepíšu návrhy další implementace aplikace.

1.1 Motivace

Hlavní motivací k výběru tohoto tématu bakalářské práce byla možnost implementovat aplikaci, která zpracovává velké množství datově obsáhlých souborů, ve kterých se dané artefakty mohou nacházet. Kromě toho to byla zvědavost, protože společnost IBM je známá. Hlavní produkty, díky kterým ji znám je hardware, jako např. mikroprocesory, nebo notebooky ThinkPad. Chtěla jsem ji tedy poznat i po softwarové stránce.

K motivaci by se dalo počítat i to, že platforma IBM Business Process Management (BPM) již vyhledávání na artefakty má, ale toto vyhledávání nemá dostatečnou funkcionalitu.

Kapitola 2
Důležité definice

V této kapitole zmíním definice, které je dobré znát pro popis platformy IBM BPM.

Nejprve si řekněme, co je to řízení procesu. Jak píše paní Svozilová ve své knize Zlepšování podnikových procesů[1]: „Řízení procesu je činnost, která využívá znalostí, schopností, metod, nástrojů a systémů k tomu, aby identifikovala, popisovala, měřila, řídila, hodnotila a zlepšovala procesy se záměrem efektivního pokrytí potřeb zákazníka procesu“.

Tímto se dostáváme k první definici pojmu business process.

2.1 Business process

Stránka techopedia [2] definuje business process jako odkaz na širokou škálu strukturovaných, často zřetězených, činností nebo úkolů. Ty jsou prováděny lidmi nebo zařízeními za účelem výroby specifické služby nebo produktu pro konkrétního zákazníka. Business procesy jsou implementovány k dosažení předem stanoveného cíle. Business procesy probíhají na všech organizačních úrovních.

Termín business proces může také odkazovat na kumulativní účinky všech kroků směřujících k business cíli. Tato posloupnost kroků může být znázorněna pomocí vývojového diagramu.

Business proces se dělí na tři základní typy. Těmi jsou:

- Provozní procesy – procesy, které představují hlavní činnost organizace. Například poskytování služeb nebo výroba specifického produktu.
- Podpůrné procesy – procesy, které podporují základní procesy. Například účetnictví a technická podpora.
- Řídící procesy – procesy, které řídí a koordinují provoz systému.

2.2 BPMN

Object Management Group [3] popisuje BPMN jako standard pro modelování business process. Tento standard umožňuje podnikům pochopit své interní

obchodní postupy v grafické notaci. Poskytuje tak organizacím možnost o nich diskutovat. Grafická notace usnadňuje pochopení spolupráce v oblasti výkonu a obchodních transakcí mezi organizacemi.

V následujících sekcích popíšu čtyři základní kategorie prvků, podle stránky omg.org [4].

2.2.1 Flow objects

Prvky toků (Flow objects) jsou základní strukturou business procesu. Prvky toků jsou tří základních typů:

- **Událost (Event)** je reprezentována kruhem a je to něco, co se stane v průběhu business procesu. Události ovlivňují průběh business procesu a obvykle mají spouštěče nebo dopad. Existují tři typy událostí podle toho, kdy ovlivňují tok: počáteční událost (start), přechodná událost (intermediate event) a koncová událost (end).
- **Aktivita (Activity)** je reprezentována obdélníkem se zaoblenými rohy. Aktivita je obecný pojem pro práci, kterou společnost provádí. Aktivita může být jednoduchá (task) nebo složená (sub-process).
- **Brána (Gateway)** je reprezentována tvarem diamantu. Brána se používá jako rozvětvení nebo slučování sekvenčního toku (sequence flow).

2.2.2 Connecting objects

Prvky toku jsou spolu propojeny v diagramu, funkci propojení poskytují tři propojovací objekty (connecting objects). Jedná se o tyto propojení:

- **Sekvenční tok (Sequence flow)** je vyobrazen plnou čarou a vyplněnou šipkou. Používá se k určení pořadí, ve kterém budou aktivity v procesu prováděny.
- **Tok zpráv (Message flow)** je vyobrazen přerušovanou čarou a nevyplněnou šipkou. Používá se pro zobrazení toku zpráv mezi dvěma samostatnými účastníky procesu, kteří je přijímají a odesílají.
- **Association (Asociace)** je vyobrazena tečkovanou a šipkou složenou z čar. Používá se k zobrazení vstupů a výstupů aktivity.

2.2.3 Swimlanes

Plavecké dráhy (swimlines) jsou mechanismus organizace činnosti do samostatných vizuálních kategorií za účelem znázornění různých funkčních schopností nebo odpovědností. BPMN podporuje plavecké dráhy se dvěma hlavními konstrukty a těmi jsou:

- **Bazén (Pool)** představuje účastníka procesu. Působí také jako grafický kontejner pro dělení souboru aktivit z jiných bazénů.

- **Dráha (Lane)** je dílčí oddíl uvnitř bazénu a rozšiřuje délku bazénu, buď svisle nebo vodorovně. Pruhy jsou používány k organizaci a kategorizaci aktivit.

2.2.4 Artifacts

Artefakty (Artifacts) rozšiřují základní notaci a poskytují schopnost dalšího kontextu vhodného pro konkrétní modelování situace. Současná verze BPMN definuje tři typy artefaktů, kterými jsou:

- **Datový objekt (Data object)** je mechanismus, který ukazuje, jaké údaje jsou vyžadovány nebo vytvářeny aktivitami.
- **Skupina (Group)** reprezentována obdélníkem vykresleným přerušovanou čarou. Skupinu lze použít pro účely dokumentace nebo analýzy, ale nemá vliv na sekvenční tok.
- **Anotace (Annotation)** je mechanismus pro poskytnutí další textové informace pro čtenáře BPMN diagramu.

Kapitola 3
Platforma IBM BPM

Tato kapitola se zabývá popisem platformy IBM BPM. Nejprve popíšu historii vzniku IBM BPM, po té přejdu k popisu IBM BPM a jeho částí, se kterými pracuji.

Zdrojem pro popis platformy IBM BPM je kniha od Neila Kolbana [5].

3.1 Historie

IBM BPM má své prvopočátky již v roce 2005, kdy společnost IBM vydala produkt WebSphere Process Server (WPS). WPS předcházel produktu BPM a jeho architektura byla orientována na obchodní služby. Tyto služby pak zákazník mohl shromažďovat, opakovaně používat a stavět obchodní řešení. Z praktického hlediska obchodní služba byla aplikací nebo funkční komponentou, která se vystavuje jako opakovaně použitelná služba. Vytvořením sady opětovně použitelných služeb, vznikne téměř finální řešení, ke kterému chybí popis řídících pravidel.

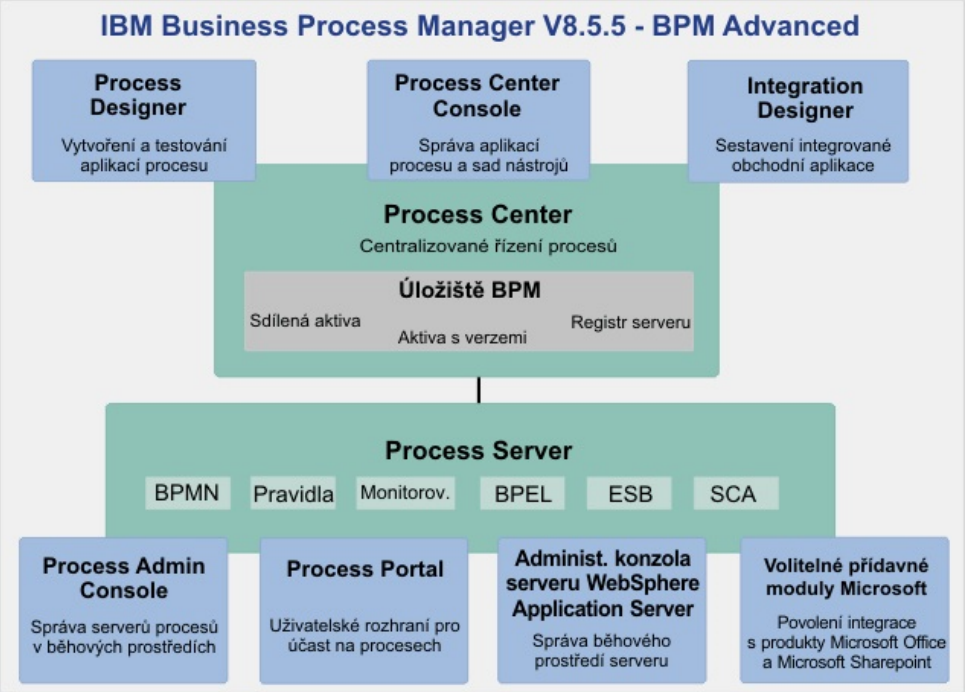
WPS poskytl implementaci pomocí otevřeného průmyslového standardu nazývaného Business Process Execution Language (BPEL). BPEL popisoval řídící pravidla, tzn. pořadí, ve kterém se kroky procesu spustí, včetně větvení, aktualizací proměnných atd. Tímto produktem IBM nepokryla veškeré možné zákazníky, protože společnost Lombardi konkurovala svým produktem TeamWorks. Tento produkt IBM získala v roce 2010 a byl označen jako WebSphere Lombardi Edition (WLE).

Až v roce 2011 společnost IBM vydala produkt BPM, nebo též IBPM, který spojil nejlepší části WPS a WLE do jedné platformy. Bylo to první vydání IBM BPM na trhu, měl označení verze 7.5 a byl zpětně kompatibilní s WPS a WLE.

Na stránkách IBM[6] lze dohledat, že současně je IBM BPM obsaženo v novém produktu jménem Business Automation Workflow (BAW). BAW kombinuje možnost správy business procesů a případů v jednom pracovním postupu.

3.2 IBM BPM

IBM BPM je komplexní nástroj používaný k modelování business procesů, na obrázku 3.1 jsou zachyceny jeho komponenty.



Obrázek 3.1: IBM BPM diagram, zdroj: [7]

IBM BPM obsahuje nástroje pro tvorbu business procesů, které lze poté automatizovat a simulovat. Po zachycení procesu je možné jej pomocí simulace otestovat pro ověření správnosti, jak podnik funguje a kam by se měl směřovat dál. Jestliže jsou některé součásti chybné, lze je změnit v reálném čase a nové řešení znovu simulovat.

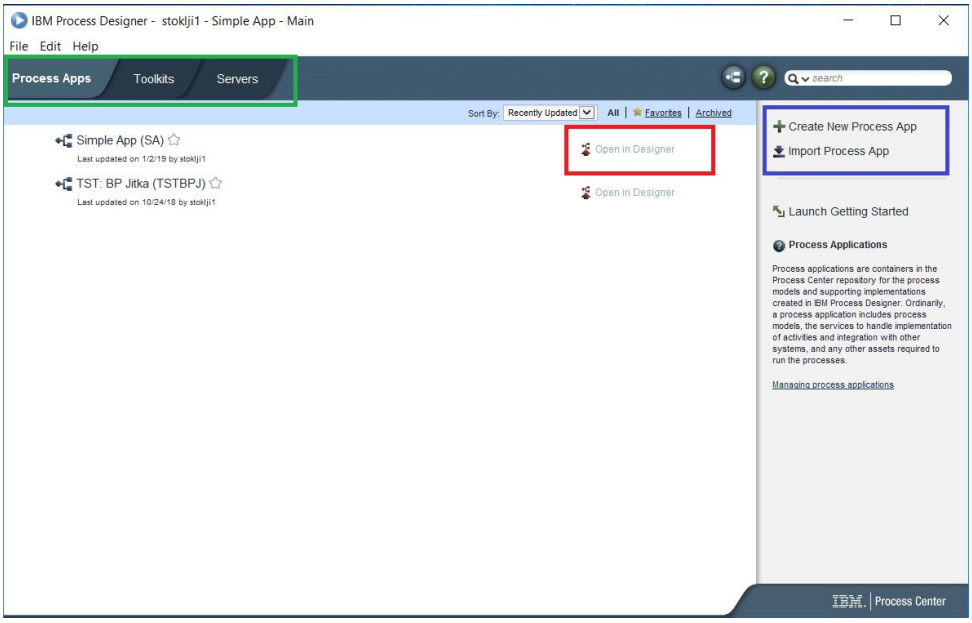
Nástroje IBM BPM, kterými lze vytvořit a spravovat business process, jsou Process Designer a Integration Designer.

3.3 Process designer

Process designer je nástroj pro navrhování, modelování a vytváření procesů. Process designer je implementován jako nadstavba pro Open Source (tzn. volně dostupný zdrojový kód) Eclipse. To znamená, že IBM BPM využívá robustnosti a známých, důvěryhodných funkcí Eclipse. Záměrně IBM BPM nevyužívá veškerých předností Eclipse, aby byl pro uživatele přívětivý a lépe pochopitelný. Process designer lze stáhnout, nainstalovat a spustit jako desktop aplikace a umožňuje popsat business process pomocí zápisu BPMN. Je též možné Process designer využívat jako webovou aplikaci. Jako součást

procesu je možné spustit externí počítačové aplikace běžící na platformě Java, které vykonají danou část procesu.

Pro možnost grafického znázornění logicky po sobě jdoucích kroků business process pomocí diagramů popsaných standardem BPMN, je nejprve nutné si vytvořit Process Application nebo Toolkit. To jsou kontejnery obsahující artefakty, dalo by se je přirovnat k projektu vytvořenému v Javě. Jak Process designer vypadá, zobrazuje obrázek 3.2.



Obrázek 3.2: Process Designer

Na obrázku 3.2 jsou viditelné tři záložky označené zeleným rámečkem. Záložky jsou pojmenovány Process Apps, Toolkits a Servers. Servery jsou dva, produkční a testovací. Produkční server obsahuje finální verze procesních aplikací a toolkitů používaných zákazníkem. Testovací server slouží pro vývoj a vyzkoušení funkčnosti nových změn před nasazením na produkční server. Pro tuto práci jsou ale zajímavé první dvě záložky. V Process Apps můžeme importovat a vytvářet procesní aplikace, tlačítka pro vytváření a import jsou na obrázku 3.2 označena modrým rámečkem, a totéž pro toolkity v druhé záložce s názvem Toolkits. Přes Open in Designer, označeným červený rámečkem, se již dostaneme do designeru a můžeme vytvářet artefakty a diagramy kroků business process.

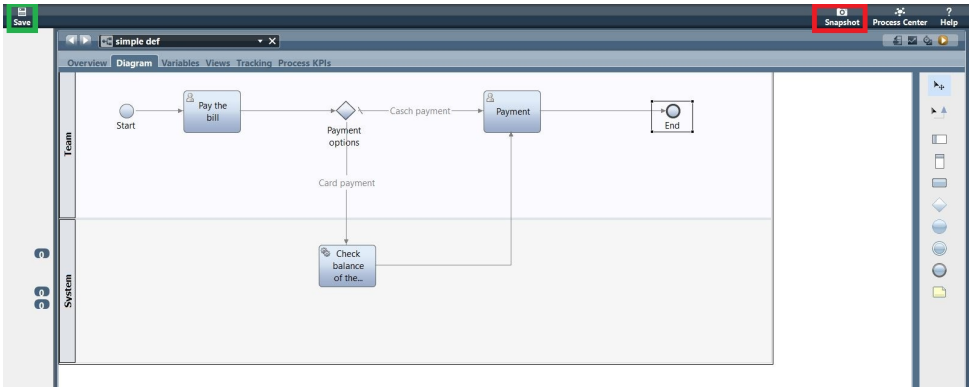
3.4 Process application

Procesní aplikace je model, nebo spíše šablona. Po jejím spuštění se vytvoří instance procesu, vždy nová při opětovném spuštění. Každá instance procesu má identifikátor a může být v jednom stavu v jedné časové jednotce. Procesní aplikace se obvykle nemaže, lze ji skrýt označením za archivovanou. Opět

zobrazit pokud je obnovena z archivu, jelikož při archivaci zůstává uložena. Jakmile je procesní aplikace v archivu, je možnost ji smazat trvale.

Procesní aplikaci můžeme vytvořit v procesním centru, které je k nalezení v process designeru webové či desktopové aplikace. Procesní centrum dále spravuje a uchovává procesní aplikace a jejich artefakty. Při uložení procesní aplikace pomocí tlačítka, označeného zeleným rámečkem na obrázku 3.3, se vytvoří tzv. Unnamed snapshot. Tlačítko na vytvoření verze procesní aplikace neboli Snapshotu, který je pojmenovaný a je možné s ním dále pracovat, je označené červeným rámečkem.

Ukázka jednoduché procesní aplikace:



Obrázek 3.3: Jednoduchá procesní aplikace

Snapshot obsahuje aktuální stav procesní aplikace, který se skládá ze všech jejích artefaktů a obsahu artefaktů. Po provedení jakékoliv změny je vhodné vytvořit další snapshot, protože provedené změny se v již uložených snapshotech neprojeví. Snapshoty v sobě nenesou informace o změnách provedených v procesní aplikaci, každý snapshot je plná kopie dat celé procesní aplikace uložená do archivu.

Procesní aplikaci lze exportovat a importovat s příponou souboru .twx. Tento soubor popíšu v podkapitole 3.6 Export a import.

3.5 Toolkit

Toolkity jsou podobné procesním aplikacím. Rozdílem je, že procesní aplikace vytváří řešení pro nasaditelnou aplikaci, což toolkit nedokáže. Toolkit je vlastně knihovna artefaktů, které lze použít ve více procesních aplikacích či jiných toolkitech. Toolkity mohou být uloženy jako systémová data. Tyto toolkity jsou označené pouze ke čtení a vytváří definici datových struktur procesních aplikací. Toolkity mohou být exportovány, importovány a vytvářeny stejně jako procesní aplikace.

Toolkitům lze také vytvářet snapshoty, jenž jsou považovány za verze. Přes snapshot pak lze přidat procesní aplikaci závislost na toolkitu.

3.6 Export a import

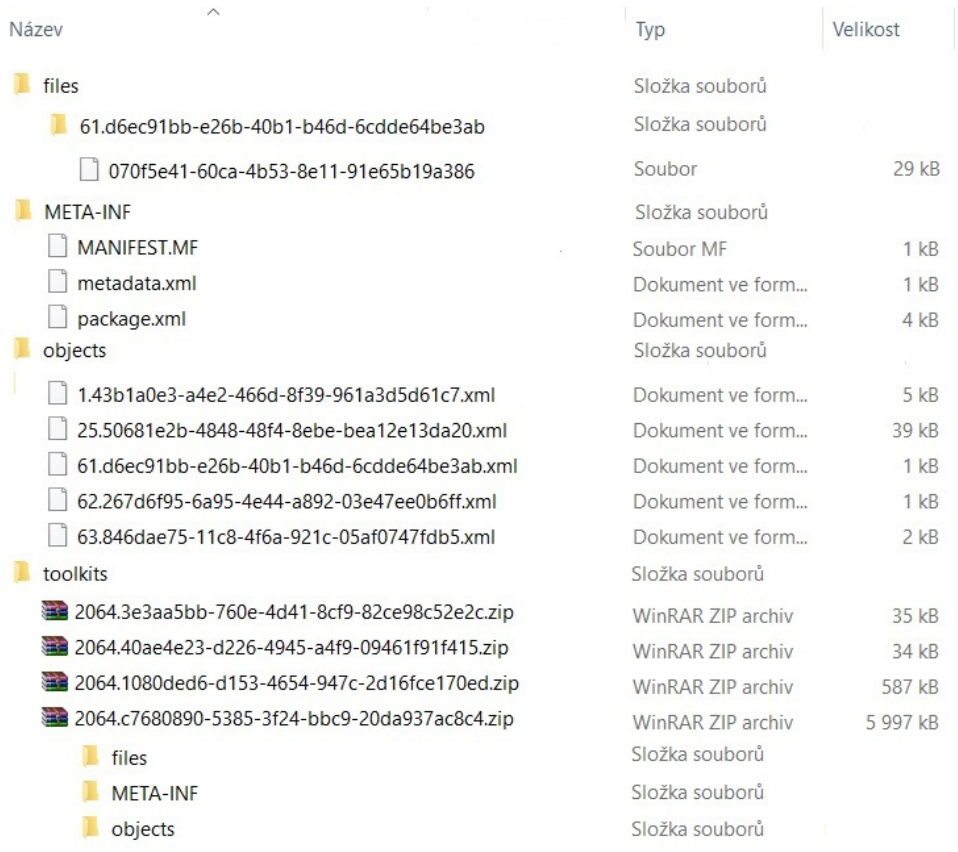
Jak již bylo zmíněno Toolkity a procesní aplikace lze z Process designeru exportovat do souboru s příponou .twx. Tento soubor je možné rozbalit pomocí aplikace na rozbalování archivů (například WinRAR). Popis adresáře po rozbalení je v kapitole 3.7. TWX soubor je možné znovu importovat do Process designeru, nebo do nástroje Integration designer.

Integration designer je založen na Eclipsové sadě nástrojů pro testování na lokální kopii procesního serveru. Tento server je komponenta, která řídí business process pomocí jeho modelu s názvem Business process definition ve formátu BPMN. Business process definition umožňuje celkový pohled na průběh procesu a jeho snadnou úpravu. Integration designer je tedy vývojovým nástrojem pro vytváření modulů a souvisejících komponent. Na moduly je nahlíženo jako na projekty.

3.7 Adresářová struktura

Po rozbalení exportovaného souboru s příponou TWX, ať už je to procesní aplikace, nebo toolkit (dále již budu psát jen o procesní aplikaci ale vztahuje se to i na toolkit), dostaneme adresář se složkami files, META-INF, objects a toolkits.

Složka toolkits obsahuje zabalené toolkity, na kterých je procesní aplikace závislá, nebo tyto toolkity závislé. Po rozbalení nalezneme podobnou adresářovou strukturu jako v původní procesní aplikaci, akorát bez složky toolkits. Tyto toolkity obsahuje složka toolkits v procesní aplikaci, kde jsou přidány všechny toolkity, na nichž jsou toolkity závislé, rekurzivně. Na obrázku 3.4 poslední tři složky jsou z rozbaleného souboru s názvem 2064.c7680890-5385-3f24-bbc9-20da937ac8c4.zip ve složce toolkits.



Obrázek 3.4: Struktura adresáře

Složka META-INF obsahuje soubor package.xml, ve kterém nalezneme jméno procesní aplikace a o jaký Snapshot se jedná. Package.xml obsahuje tag `<project>` a v něm atribut „isToolkit“, který je v případě toolkitu nastaven na hodnotu „true“ a v případě procesní aplikace na hodnotu „false“. Dále zde můžeme naleznout seznam toolkitů, na kterých je procesní aplikace závislá, ve složce toolkits. K nalezení je také seznam souborů ze složky files, tyto soubory jsou zahrnuty do procesní aplikace nebo toolkitu, ale nebyly vytvořeny v Process designeru. Ve složce objects nalezneme seznam artefaktů použitých v procesní aplikaci.

3.7.1 Artefakt

Procesní artefakt je XML soubor popsaný ve standardu BPMN, který reprezentuje objekty vytvořené v toolkitu nebo procesní aplikaci. Celý artefakt popsaný v XML souboru je velký, proto obrázek 3.5 ukazuje jen část artefaktu.

Obrázek 3.5: Struktura artefaktu business process definition

Pro tuto práci, a to vyhledávání artefaktů, je důležitý řádek `<bpd id="25.50681e2b-4848-48f4-8ebe-bea12e13da20" name="simple def">`, tento celý řetězec id je unikátní v celé procesní aplikaci, či toolkitu. První číslo před tečkou říká jakého typu artefakt je, v tomto případě je to číslo 25 a říká, že artefaktem je proces. Zbylý řetězec za tečkou je id vygenerované ke jménu artefaktu. Artefakt může mít stejný název jen v případě, že je jiného typu. Například nemůže existovat další artefakt se jménem „simple def“, který by byl procesem, ale pokud bychom vytvářeli artefakt datového typu, je možné ho znovu pojmenovat „simple def“.

Kdybychom se snažili napsat to, co je popsáno v XML artefaktem, bylo by to podobné třídě v Javě. Třída v Javě je přímou reprezentací, zatímco artefakt je abstrakcí těchto dat.

Typy artefaktů, kterými se zabývá tato práce:

- Processes – modely business procesu, jsou základním stavebním kamenem, nabývají těchto typů:
  - Business Process Definition
  - Process
- User interface – služby používané k interakci s člověkem, jsou jimi:
  - Ajax Service
  - Client-side Human Service
  - Heritage Human Service
- Services – služby poskytující opětovně použitelnou implementaci jednotlivých kroků, patří mezi ně:
  - Decision Service
  - General System Service
  - Service Flow

- Undercover Agent – poslouchá aktivační události z externích systémů a spouští instance procesu přímo nebo pomocí počáteční události procesu
- Team – skupina lidí, která vykonává akce procesu
- Business Object – datový typ používaný k popisu sady atributů, se kterými proces pracuje

Teď již víme, že artefakty mají podobu XML souborů, možnosti vyhledávání v XML souborech popisuji v kapitole 4.1.

Kapitola 4

Metody řešení vyhledávání v XML souborech a vizualizace nalezených diagramů

V této kapitole jsou popsány možnosti vyhledávání v XML souborech a vizualizace nalezených diagramů obsahujících hledaný artefakt. Do IBM BPM lze zakomponovat soubory implementované v Javě a zkompilované do souboru s příponou .jar. Jak se můžeme dozvědět na stránce ReviverSoft[8] JAR je formát pro soubory v Java archivu. V souboru JAR jsou aplikační knihovny a referenční údaje pro platformy, ve kterých je Java povolena. JAR soubor zabaluje celý kód psaný v Javě a vytváří spustitelnou aplikaci. IBM BPM běží na platformě Java, proto vyhledávání v XML a vizualizace nalezených diagramů řeším v Javě, aby použití a případná integrace do IBM BPM byla co nejjednodušší.

4.1 Vyhledávání v XML souborech

V jazyce Java lze vyhledávat v XML souborech dvěma způsoby. Jako stromovou strukturu, nebo stream. Soubory XML jako stromovou strukturu analyzuje DOM a XPath, jako stream dat jej zpracovává SAX, StAX. Jednotlivé metody popíšu v následujících sekcích, v poslední sekci napíšu vybranou metodu vyhledávání v XML souborech a důvod výběru. Zdrojem pro popis těchto metod je stránka tutorialspoint[9].

4.1.1 Document Object Model (DOM)

DOM definuje rozhraní, které programům umožňuje přístup a aktualizaci stylu, struktury a obsahu XML souborů. DOM se používá:

- je-li nutná vysoká znalost struktury XML souboru
- je-li potřeba přesouvat části XML souboru
- je-li informace použita v XML souboru více než jednou.

Při analýze XML souboru pomocí analyzátoru DOM získáme strukturu stromu, který obsahuje všechny prvky dokumentu. DOM poskytuje řadu funkcí na prozkoumání obsahu a struktury XML souboru.

Výhodou je, že DOM je společné rozhraní pro manipulaci se strukturami souborů. Jedním z cílů DOM je kompatibilita analyzátorů psaných ve standardu DOM bez úprav kódu.

4.1.2 Simple API for XML (SAX)

SAX je analyzátor založený na událostech pro XML soubory. SAX je streamovací rozhraní pro XML, tzn. že aplikace používající SAX obdrží upozornění na události v XML souboru, který je zpracováván jako prvek a atribut najednou v sekvenčním pořadí. SAX se používá:

- je-li možné zpracovávat XML soubor lineárně
- je-li strom XML souboru náročný na paměť
- zahrnuje-li problém k vyřešení pouze část XML souboru

Výhodou je použití při zpracovávání XML souboru přes stream, protože SAX čte soubor shora dolů.

Nevýhodou je, že nemáme libovolný přístup k XML souboru, protože je zpracováván dopředu. Pokud potřebujeme sledovat data zpracovaná analyzátorem, nebo změnit pořadí položek, musíme napsat kód a uložit data na vlastní úložiště.

4.1.3 JDOM

JDOM je Open Source knihovna založená na jazyce Java, která analyzuje XML soubory. Je to Java API pro vývojáře a je optimalizována pro Javu. JDOM je kombinací rozhraní DOM a SAX API. Pracuje s malým množstvím paměti a je téměř rychlá jako SAX.

Použití JDOM je stejné jako použití DOM. Výhodami JDOM jsou rychlost, poskytnutí flexibility pro vývojáře v Javě a snadná údržba analytického kódu XML.

4.1.4 Streaming API for XML (StAX)

StAX provádí podobnou analýzu XML souboru jako SAX. Rozdílem je, že aplikace žádá StAX o získání informací z XML, zatímco SAX upozorňuje klientskou aplikaci na získané informace z XML. StAX dokáže číst i psát XML soubory, SAXem jej lze pouze číst.

StAX má stejné použití, výhody a nevýhody jako SAX.

4.1.5 XML Path Language (XPath)

XPath definuje jazyk pro nalezení informací v XML souboru. XPath poskytuje různé typy výrazů, které lze použít k dotazování na relevantní informace z

XML souboru. XPath je založen na stromové reprezentaci XML souboru, umožňuje procházet strom a vybírat uzly vyhovující různým kritériím.

4.1.6 DOM4J

DOM4J je Open Source knihovna založená na jazyce Java stejně jako JDOM. Je to flexibilní a paměťově efektivní API, optimalizované v jazyce Java. DOM4J pracuje s DOM, SAX, XPath a Extensible Stylesheet Language Transformations (XSLT). Může analyzovat velké XML soubory s malou náročností na paměť. XSLT je jazyk transformující soubory XML do jiných souborů.

DOM4J má stejné využití jako DOM a stejnou výhodu jako JDOM.

4.1.7 Vybraná metoda vyhledávání v XML souborech

Na parsování a poté vyhledávání v souborech s příponou .xml jsem použila metodu SAX.

XML soubory stačí zpracovávat lineárně. Pro parsování a poté uložení dat na vizualizaci nalezených diagramů není nutné procházet celý soubor. Stromová struktura XML souboru popisujícího artefakt je obvykle náročná na paměť.

Tato technologie parsování XML souborů se tedy jeví jako vhodná pro můj případ využití.

4.2 Vizualizace nalezených diagramů

Na vizualizaci nalezených diagramů je potřeba vytvořit Graphical User Interface (GUI). GUI lze v Javě vytvořit pomocí Abstract Window Toolkit (AWT), Swing, nebo JavaFX.

Stránka studytonight[10] uvádí, že AWT obsahuje sadu nástrojů pro návrh GUI, které by mohlo pracovat na různých platformách. AWT je platformě závislá, GUI navržené na jedné platformě může na jiné platformě vypadat jinak. AWT je základem pro Swing.

Swing je sada rozhraní GUI, která rozšiřuje AWT. Na rozdíl od AWT, jak se můžeme dozvědět ze stránky javatpoint[11], je Swing platformě nezávislá, dodržuje zásady Model View Controller (MVC) a poskytuje součásti jako jsou tabulky, seznamy apod.

Na stránce ITnetwork[12] se píše, že JavaFX by v budoucnu měla nahradit Swing. JavaFX oproti Swingu podporuje obrázky, videa, hudbu, grafy, CSS styly a další technologie. JavaFX by měla být jednoduchá na použití a je vhodná jak pro desktopové aplikace, tak pro webové či mobilní aplikace. V JavaFX lze vyvíjet podobně jako ve Swingu, nebo použitím FXML. FXML je jazyk pro návrh formulářů odvozený z XML.

4.2.1 Vybraná technologie na vizualizaci nalezených diagramů

Na vizualizaci nalezených diagramů jsem vybrala technologii Swing. Důvody jsou, že jsem s ním již pracovala v jednom předmětu při psaní semestrální práce a zároveň splňuje veškeré požadavky na vizualizaci diagramů.

Během psaní implementační části práce jsem konzultovala s Bc. Annou Márií Hriadelovou vykreslování diagramů [13].

Kapitola 5

Návrh implementace

V této kapitole popíšu návrh implementační části bakalářské práce. Nejprve vypíšu funkční a nefunkční požadavky na aplikaci, ty dále specifikuji pomocí případů užití. Poté popíšu diagram aktivit a UML diagram tříd.

5.1 Funkční a nefunkční požadavky aplikace

V této sekci uvedu funkční požadavky a nefunkční požadavky na aplikaci, které jsem definovala na základě zadání bakalářské práce, jenž bylo dále upřesněno odborným konzultantem.

Funkčními požadavky aplikace jsou:

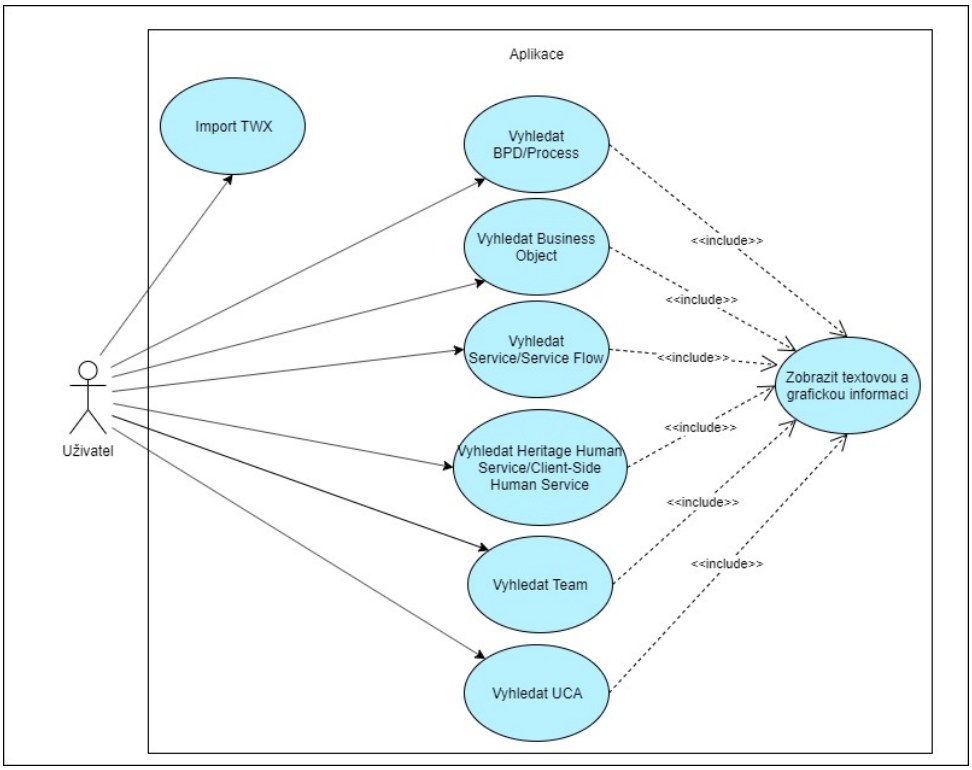
- F1 – Import TWX souboru
  Aplikace musí umožnit import souboru s příponou TWX.
- F2 – Vybrání artefaktu
  Aplikace musí umožnit vybrat ze seznamu artefaktů obsažených v souboru s příponou TWX.
- F3 – Vyhledání artefaktu
  Aplikace musí vyhledat použití artefaktu v ostatních artefaktech (pro předem zadané případy).
- F4 – Zobrazit textovou informaci o artefaktu
  Aplikace musí zobrazit textovou informaci o použití artefaktu v jiném artefaktu.
- F5 – Zobrazit grafickou informaci o artefaktu
  Aplikace musí zobrazit diagramy, ve kterých je artefakt použit.

Nefunkčními požadavky aplikace jsou:

- NF1 – Desktop aplikace s grafickým rozhraním
  Samostatně spustitelná desktopová aplikace bez závislosti na serverové či databázové straně s grafickým uživatelským rozhraním.
- NF2 – Aplikace vyvíjená v jazyce Java
  Aplikace musí být vyvíjena v jazyce Java.

5.2 Use Cases

Případy užití aplikace, neboli Use Cases, dále specifikují funkční požadavky na aplikaci. V případě této aplikace je aktér jen uživatel, jenž ji používá. Případy užití znázorňuje následující obrázek 5.1.

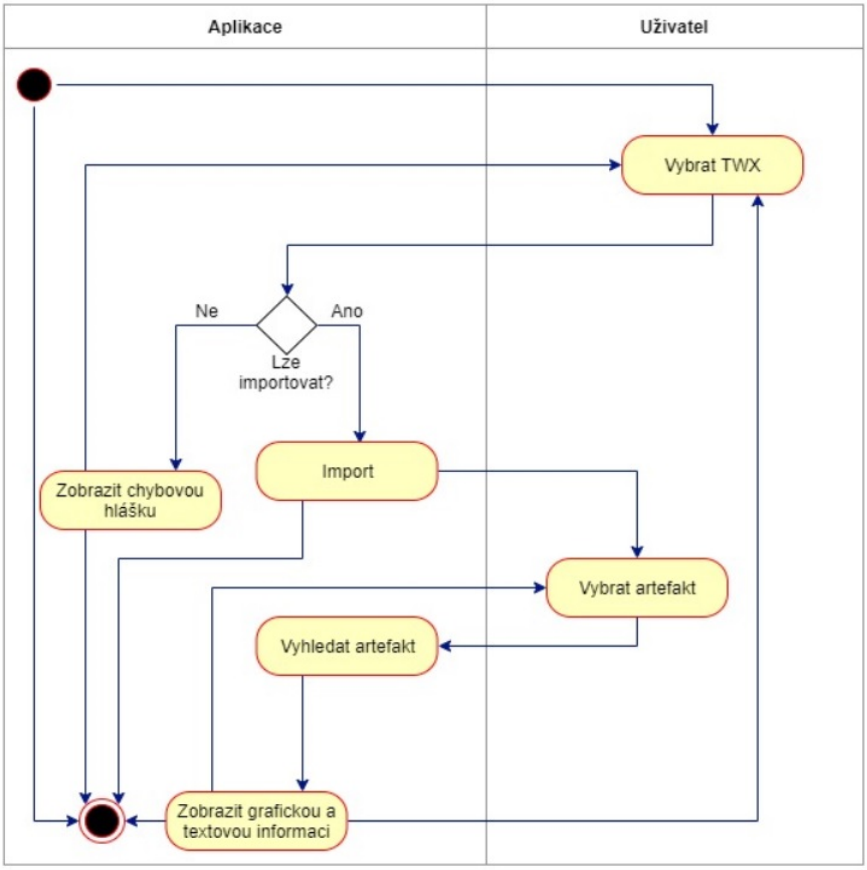


Obrázek 5.1: Případy užití aplikace

Případy užití rozdělíme na tři kategorie, a to:
Import TWX – aplikace importuje soubor s příponou .twx a nalezne všechny artefakty.
Vyhledávání artefaktů – aplikace vyhledá vybraný artefakt v diagramech ze souboru s příponou .twx.
Zobrazení textové a grafické informace – aplikace zobrazí textovou informaci o nepoužití artefaktu v případě, kdy artefakt není nalezen v nějakém diagramu. Aplikace zobrazí textovou informaci v jakém diagramu je vybraný artefakt použit a vykreslí diagram s označením použití vybraného artefaktu.

5.3 Diagram aktivit

Na diagramu aktivit, obrázek 5.2, lze vidět průchody aplikací.



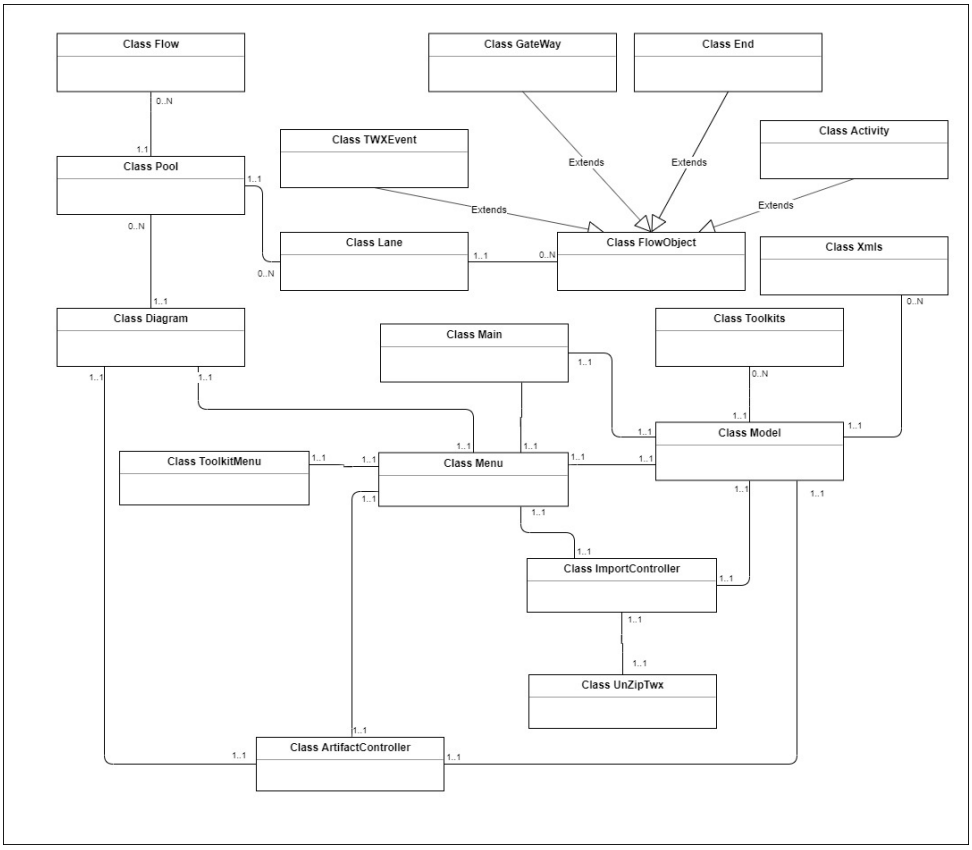
Obrázek 5.2: Diagram aktivit

Pokud je vybrán soubor, který nemá příponu .twx, je zobrazena chybová hláška a uživatel může znovu vybírat soubor. Při správném importu může uživatel vybrat artefakt a aplikace poté zobrazí výskyt artefaktu a vykreslí k tomu diagram. V případě více výskytů vykreslí nalezené diagramy s jejich názvy pod sebou.

5.3.1 UML diagram tříd

Pro lepší čitelnost a možnost rozšiřitelnosti kódu v mé bakalářské práci používám návrhový vzor MVC, neboli model-view-controller. Tento návrhový vzor rozděluje aplikaci na tři samostatné sekce.

Model v sobě ukládá datové typy a informace, view se stará o grafické zobrazení dat uživateli a controller zajišťuje uložení změn provedených pomocí view do modelu. Toto s sebou nese i rozdělení tříd do těchto sekcí. Obrázek 5.3 zobrazuje UML diagram tříd vyvíjené aplikace.



Obrázek 5.3: UML diagram tříd

Popis tříd z obrázku:

- Main – třída obsahující spouštěcí metodu
- FlowObject – abstraktní modelová třída pro artefakty
  - End – modelová třída pro koncový artefakt
  - Activity – modelová třída pro artefakt ve tvaru obdélníku
  - GateWay – modelová třída pro artefakt ve tvaru diamantu
  - TWXEvent – modelová třída pro artefakt ve tvaru kruhu
- Lane – modelová třída pro swimline v procesu
- Xmls – modelová třída pro data z XML souborů
- Flow – modelová třída pro šipky spojující artefakty
- Pool – modelová třída pro diagram
- Toolkits – modelová třída pro toolkity
- ToolkitMenu – modelová třída pro názvy záložek jak menu tak i toolkitu

- Model – modelová třída pro vytvoření instancí na veškeré modelové třídy a jednodušší komunikaci s třídou Diagram
- Menu – třída na vykreslení menu na import a menu na výběr artefaktů
- Diagram – třída na vykreslení nalezených diagramů, kde je artefakt použit
- ArtifactController – třída obsluhující vyhledávání vybraného artefaktu
- ImportController – třída obsluhující import snapshotu
- UnZipTwx – třída na rozzipování TWX souboru

Kapitola 6

Implementace

V této kapitole popíšu implementační část bakalářské práce. Rozdělení popisu je stejné, jako rozdělení případů užití v kapitole 5.2. Tedy rozdělím popis na import TWX, vyhledávání artefaktů a zobrazení textové a grafické informace o použití artefaktu. Aplikace je naprogramovaná v jazyce Java za použití knihovny Swing na vykreslení diagramů a knihovny SAX na parsování XML souborů. Nejprve pro úplnost zmíním software, ve kterém jsem vyvíjela aplikaci, aby bylo v budoucnu možné jej napodobit pro navázání další implementace.

6.1 Specifikace vývojového prostředí a spuštění aplikace

Aplikaci jsem vyvíjela v:

- operační systém – Windows 10 Home verze 1903
- vývojové prostředí – JetBrains Intellij IDEA verze 19.2.3 Ultimate Edition
- jazyk – Java verze 1.8
- nástroj pro správu a automatizaci buildů – Apache Maven verze 3.6.0
- prostředí pro spuštění JAR souboru – JRE verze 1.8

S vývojem souvisí také kompilace a spuštění aplikace. Pro kompilaci a spuštění aplikace je podmínkou mít v systému nainstalovanou Javu verze 1.8, Apache Maven verze 3.6.0 a JRE verze 1.8. Aplikaci lze kompilovat a spustit dvěma způsoby, těmi jsou:

- Prvním způsobem je build přes vývojového prostředí. Je nutné nastavit main třídu, ta je k nalezení v `src/main/java/Main/Main.java`. Poté stačí spustit `build` a `run`. Je-li k dispozici vývojové prostředí, které umí zacházet s pluginy Apache Maven, jako další možnost je spustit v Apache Maven plugin `package`. Vytvoří se složka target, ve které, mimo jiné, naleznete soubor `twxsearch-1.0-SNAPSHOT-jar-with-dependencies.jar`. Pak stačí dvojklikem spustit JAR soubor.

- Build a spuštění lze také provést v příkazové řádce. Ve složce twxsearch spustíte příkaz `mvn package`, poté přejdete do složky target a spustíte aplikaci pomocí příkazu
  `java -jar twxsearch-1.0-SNAPSHOT-jar-with-dependencies`.

Nyní se přesunu k popisu implementace.

6.2 Import TWX

Aplikace pro spuštění nepotřebuje žádné atributy. Po spuštění aplikace se načte okno grafického uživatelského rozhraní jako na obrázku 6.1.

Obrázek 6.1: Okno grafického uživatelského rozhraní

Aplikaci po spuštění lze pouze vypnout křížkem v pravém horním rohu, na obrázku 6.1 označeno červeným rámečkem, nebo importovat TWX soubor. V levém horním rohu lze nalézt kartu **File** označenou modrým rámečkem, po jejím rozkliknutí nalezneme položku **Import**. Pro importování TWX stačí na tuto položku kliknout.

Po kliknutí na položku import se vytvoří událost z uživatelského rozhraní, kterou odchytí a obslouží ImportController pomocí kódu na následujícím obrázku 6.2.

```
public void actionPerformed(ActionEvent e) {
    menu.getLabel().setVisible(false);
    menu.getCanvas().setVisible(false);
    String path = menu.showFileOpen();
    menu.clearBar();
    if(path != null){
        if(!check(path)){
            menu.showError();
        }
        updateArtifacts();
        menu.loadBar();
    }
}
```

Obrázek 6.2: ImportController.java

Jelikož předpokladem je, že se může import zavolat i po vyhledání použití artefaktu, tak první provedená akce zneviditelní nalezené diagramy a informace o posledním hledaném artefaktu. Dále ImportController pomocí metody showFileOpen() vyvolá JFileChooser, což je průzkumník souborů. A uživatel vybere umístění snapshotu k importu. ImportController vyčistí levé menu, kde lze vybrat artefakt k nalezení, vyzkouší v metodě check() validitu vybrané cesty a snapshotu. Při neexistenci cesty či vybrání souboru, který není TWX, vyskočí okénko s ohlášením erroru pomocí metody showError(). Jinak se provede rozzipování souboru do adresářové struktury, kterou má v sobě snapshot a která byla popsána v kapitole 3.7.

Pak se načtou potřebné informace z XML souborů na rozřazení a uložení artefaktů do modelu aplikace. V modelu jsou artefakty uloženy samostatně pro service, procesy a ostatní artefakty. Všechny artefakty, které začínají `25.<idArtefaktu>`, se uloží jako procesy, pro identifikaci procesu musíme otevřít XML soubor a najít tag `<type>` s číselným údajem. Business process definition má psán tag `<type isNull=true/>`, jenž si pro zjednodušení označím jako 0. Process má v tagu `<type>1</type>`, který mám samozřejmě jako 1, tímto způsobem rozlišuji process od business process definition. Podobným způsobem rozlišuji service, které začínají `1.<idArtefaktu>`. Service v sobě také obsahují tag `<type>` s hodnotou od 1 do 12. Ostatní artefakty mají unikátní počátek před id artefaktu.

Po správném rozřazení na uložení do modelu, si díky tomu aplikace přiřadí názvy artefaktů do menu v levé části aplikace, na obrázku 6.1 označené zeleným rámečkem. V této části lze vybrat artefakt na vyhledávání do 10 sekund od zadání snapshotu do importu. Zde se dostáváme k další části popsané v kapitole 6.3, kde je vyhledávání artefaktů popsáno.

6.3 Vyhledávání artefaktů

Vyhledávání probíhá podle případů užití. Artefakty mají specifikováno v jakém diagramu se mohou vyskytovat. Po vybrání artefaktu v levém menu se vytvoří akce, kterou odchytí a obslouží ArtifactController.java. Ten si z události vybere, který artefakt ho zavolal. Podle typu artefaktu je zavolána metoda se stejným jménem jako je typ artefaktu. A v ní se dále rozvětvuje hledání ve specifikovaných diagramech. Pro hledání v diagramech je nutné znovu otevírat XML soubory, což provede kód na následujícím obrázku 6.3.

```
private Element openFile(Xmls xml) throws JDOMException, IOException {
    File inputFile = new File( pathname: UNZIPOBJECTS + xml.getBox() + DOT + xml.getId() + XML);
    SAXBuilder dBuilder = new SAXBuilder();
    Document doc = dBuilder.build(inputFile);
    return doc.getRootElement();
}
```

Obrázek 6.3: Otevření XML souboru

Tato metoda otevře správný XML soubor ze složky Objects podle typu a id artefaktu. Poté pomocí SAXBuilderu naparsuje XML soubor na document a vezme jeho root element, to je párový tag, obaluje veškerý obsah XML souboru. Root element ve všech artefaktech je `<teamworks>`.

Díky root elementu jdou filtrovat specifické tagy. Aplikace vyfiltruje tagy, ve kterých se nachází napojení na artefakty. Takové tagy většinou začínají `<attached*>`, nebo končí jako `<*Ref>`, samozřejmě to může být kombinace obojího. Pokud aplikace narazí na id hledaného artefaktu, tak se dělí vyhledávání artefaktu v XML souboru, podle toho jakého typu rozparsovaný XML souboru je, na tři možnosti.

První možností je vyhledávání v XML souboru business process definition nebo process. V tomto případě pro získání diagramu stačí zkoumat, co se vyskytuje v tagu `<BusinessProcessDefinition>`. Čáry lze nalézt pod tagem `<flow>` a objekty diagramu pod tagem `<flowObject>`.

Druhou možností jsou XML soubory pro service. V takovém diagramu jsou všechny důležité informace na vykreslení v tagu `<process>`. Z něj aplikace vytáhne objekty diagramu, které se nacházejí v tagu `<item>`, čáry nacházející se v tagu `<link>` a start v tagu `<startPoint>`.

Poslední možností je samostatné vyhledávání v client-side human service. Client-side human service má stejnou strukturu jako service. Ovšem informace na vykreslení diagramu jsou více zanořené a tedy tag, ve kterém jsou k nalezení tyto informace, je `<coachflow>`. Tady je tagů, ve kterých se nacházejí objekty, mnohem více, proto jsou uvedeny jen příklady tagů. Pro start a end diagramu to jsou tagy `<ns17:startEvent>` a `<ns17:endEvent>`, pak pro čáry diagramu to je tag `<ns17:sequenceFlow>`.

Ze všech třech možností vytáhne aplikace informace. Odkud a kam vedou čáry, souřadnice a typ objektu a zjistí, zda-li to není hledaný artefakt. To vše se uloží do modelu. ArtifactController ještě zkoumá, jestli je nalezen alespoň jeden diagram. Pokud ne, vypíše se hláška „diagram není nikde použit“ jako

na obrázku 7.14 z testovacích scénářů. Na druhou stranu, diagramů může být více. Ovšem to už se dostáváme ke kapitole 6.4, vykreslování diagramu.

6.4 Zobrazení textové a grafické informace

Jakmile má aplikace uloženy informace o diagramech, kde se vyhledaný artefakt nachází v modelu, přichází na řadu vykreslování. Vykreslování obsluhuje třída Diagram.java, která prvně vykreslí název diagramu a pak zavolá u všech objektů a čar metodu na vykreslení.

Objekt ví, jakého je typu, tedy ví, jestli vykreslit kolečko, obdélník nebo diamant. Také zná své souřadnice, kde se promítne a barvu, se kterou se má vykreslit. Pokud je to hledaný artefakt, pak se sám označí zeleným rámečkem, jako na obrázku 6.4, který vykresluje obdélník.

```
int y = getY() + getHeightOfLane();
graphics2D.setColor(getColor());
graphics2D.fillRect(getX(), y, getWidth(), getHeight());
if (isSelected()) {
    graphics2D.setColor(new Color( r: 19,  g: 128,  b: 11));
    graphics2D.setStroke(new BasicStroke( width: 5));
} else {
    graphics2D.setColor(BLACK);
}
graphics2D.drawRect(getX(), y, getWidth(), getHeight());
```

Obrázek 6.4: Vykreslení obdélníčku

Zeleným rámečkem se neoznačí vyhledání týmu v heritage human service a client-side human service. A také při vypsání použití business object v jiném business object, jelikož tyto objekty se chovají jako proměnná a nemají diagram.

Při vykreslování čar, mají tyto čáry v sobě uložené odkud a kam vedou, mají tedy souřadnice levého horního rohu objektů. Čáry v sobě obsahují výpočet, který určí přesné souřadnice kde končí. Na jednom z konců je dokreslený trojúhelník, který naznačuje šipku a tedy i směr čáry. Dále se v čárách počítá zalamování.

Začíná to tím, že aplikace ví do jakého portu vede čára, port vytáhne z XML souboru. Portem je například `rightCenter`, který říká, že čára vede do objektu vpravo uprostřed. Tím vypočítá přesné souřadnice pro čáru. Aplikace má zadefinovanou vzdálenost, kterou musí čára být vykreslena směrem z portu. Poté ještě vypočítá vektory, které určují nejbližší směr k druhému konci a vypočítá to pro oba konce čáry. Od těchto souřadnic probíhá výpočet pomocí lineární algebry.

Výpočet začíná tím, že aplikace zjistí, jestli mají čáry stejnou x nebo y souřadnici. Pokud ano, vykreslí se rovná čára mezi těmito souřadnicemi. Pokud ne, přes skalární součin spočítá, jestli se čáry protnou. V případě že ano, tak se podle směru obou vektorů dopočítá bod, kde se spojí a vykreslí dvě čáry. Toto zajistí, že celek bude vypadat jako lomená čára. Poslední možnost,

kdy se tyto čáry nespojí je, že oba vektory jdou proti sobě, ale neprotly by se. Tedy díky těmto vektorům se aplikace výpočtem dostane k dvěma bodům zlomu, které budou uprostřed vzdálenosti a stačí přes ně vykreslit tři čáry. Po vykreslení tyto tři čáry vypadají jako dvakrát lomená čára. Vykreslování zalomených čar provádí aplikace pomocí metody drawPolyline(). Část výpočtu pro dvakrát lomenou čáru a volání metody je viditelné na obrázku 6.5.

```
//drawing line with two breaks
if(resultVectorFrom[0] == 0){
    half = xyfrom[1]-xyto[1];
    if(half<0){
        half*=-1;
    }
    half/=2;
    int y = xyfrom[1] + half*resultVectorFrom[1];
    graphics.drawPolyline(new int[] {flow.getXYFrom()[0], xyfrom[0], xyfrom[0], xyto[0], xyto[0],
            flow.getXYTo()[0]}, new int[] {flow.getXYFrom()[1], xyfrom[1], y, y, xyto[1], flow.getXYTo()[1]}, 6);
```

Obrázek 6.5: Zalomení čáry

Nalezených použití může být více, tento cyklus vykreslení se koná tolikrát, kolik je nalezeno použití daného artefaktu.

Po vykreslení je možné buď hledat jiný artefakt. Nebo provést znovu import nějakého snapshotu, poté vyhledávat artefakty nebo ukončit aplikaci.

Tímto byla implementace dokončena a je možné přejít otestování vyvinuté aplikace.

Kapitola 7
Testovaní

Na otestování aplikace jsem využila Unit testy, testovací scénáře a integrační testy napsané podle testovacích scénářů.

7.1 Unit testy

Při vyvíjení aplikace je možné zavléct do kódu chyby, tomu se říká regrese. Podchytit tyto chyby může regresní testování, které testuje po přidání nové funkcionality do kódu, že se v nově napsaném kódu nevyskytují chyby.

Do regresního testování patří Unit testy, pro Javu jsou to knihovny JUnit a TestNG. Vybrala jsem knihovnu TestNG z důvodu, že má více anotací a proto se lépe píší integrační testy aplikace.

Unit testy jsem psala na public metody všech tříd aplikace, mimo abstraktní třídy FlowObject. Při testování jsem dodržovala strukturu vytvořených adresářů, v případě Javy jsou to balíčky, a třídy v nich.

7.2 Vstupní podmínky

Vstupními podmínkami pro spuštění aplikace, je mít nainstalované Java Runtime Environment, neboli JRE, verze 1.8 a vyšší. V takovém případě lze spustit aplikaci dvojklikem na twxSearch.jar v přiloženém CD. Ovšem pro vyzkoušení aplikace, je nutné mít připravený snapshot procesní aplikace vytvořené v Process Designeru.

7.3 Testovací scénáře

Sepsala jsem 6 testovacích scénářů, které jsou popsány v následujících sekcích 7.3.1 až 7.3.6, na pokrytí veškerých průchodů aplikací. Průchody aplikací jsou definované pomocí diagramu aktivit, jenž zobrazoval obrázek 5.2. Ve všech testovacích scénářích, které mají vykreslit více diagramů, jsem byla nucena udělat screenshot pro tyto diagramy samostatně. Důvodem je, že kvůli množství a velikosti diagramů nelze vše zachytit pomocí jediného screenshotu.

Podobnosti vyhledaných diagramů, jenž vykresluje aplikace, a vytvořených diagramů v process designeru by přidalo mnoho obrázků. Na porovnání

jsem přidala do testovacího scénáře 1 za skutečné výstupy obrázky diagramů v proces designeru. Veškeré obrázky nalezených diagramů aplikací a diagramů vytvořených v process designeru jsou k nalezení na CD v souboru porovnaniDiagramu.pdf.

7.3.1 Testovací scénář 1

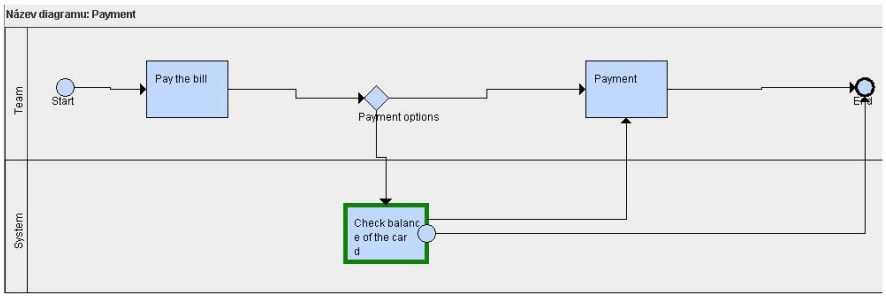
Vyhledání Business Process Definition a Process:

1. Spusťte aplikaci.
2. Naimportujte přiložený scenar1.twx.
3. Vyhledejte Business Process Definiton s názvem „Check balance of the card“.

   Očekávaný výstup – jeden diagram:

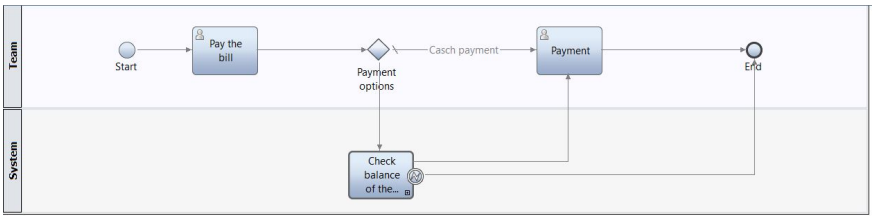
   Vykreslí se diagram s názvem „Payment“ a označeným artefaktem „Check balance of the card“ zeleným rámečkem.

   Skutečný výstup:



Obrázek 7.1: Testovací scénář 1 – nalezení business process definition v jiné business process definition

Diagram vytvořený v process designeru k porovnání



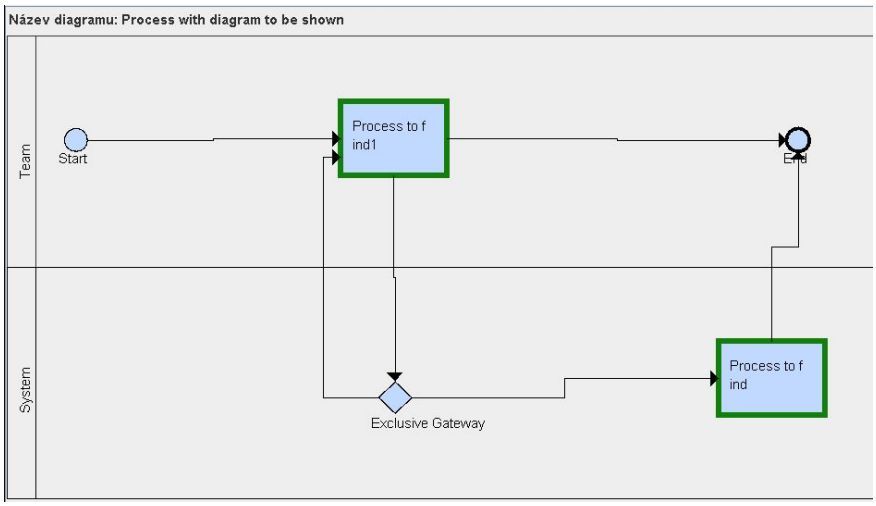
Obrázek 7.2: Process designer – diagram s názvem Payment

4. Vyhledejte Process s názvem „Process to find“.

   Očekávaný výstup – jeden diagram:

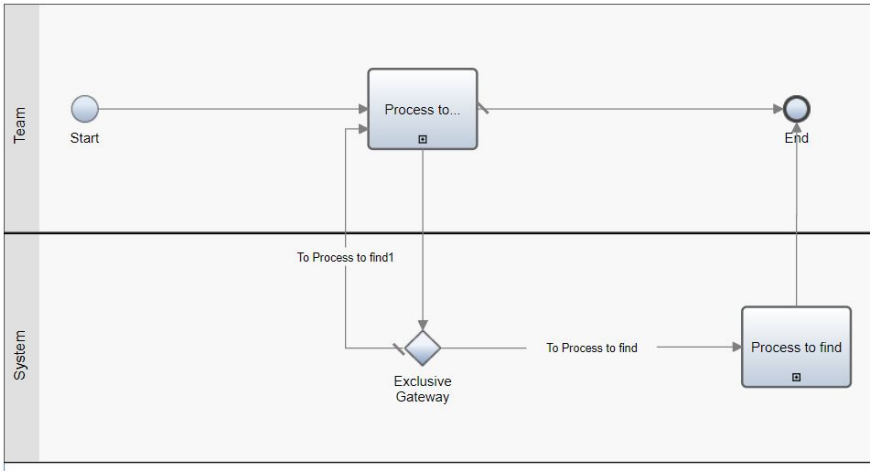
   Vykreslí se diagram s názvem „Process with diagram to be shown“ a označenými artefakty „Process to find“ a „Process to find1“ zelenými rámečky.

Skutečný výstup:



Obrázek 7.3: Testovací scénář 1 – nalezení process v jiném process

Diagram vytvořený v process designeru k porovnání



Obrázek 7.4: Process designer – diagram s názvem Process with diagram to be shown

5. Ukončete aplikaci křížkem v pravém horním rohu.

7.3.2 Testovací scénář 2

Vyhledání UnderCover Agent a import souboru, který není TWX.

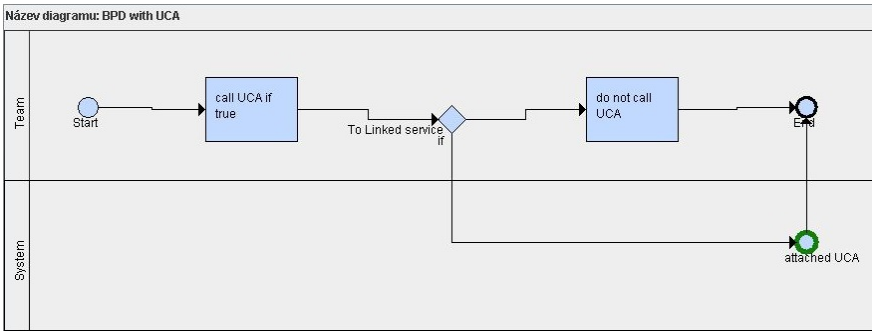
1. Spusťte aplikaci.
2. Naimportujte přiložený scenar2.twx.

3. Vyhledejte UnderCover Agent s názvem „UCA“.

Očekávaný výstup – čtyři diagramy:

Vykreslí se diagram s názvem „BDP with UCA“ a označeným artefaktem „attached UCA“ zeleným rámečkem.

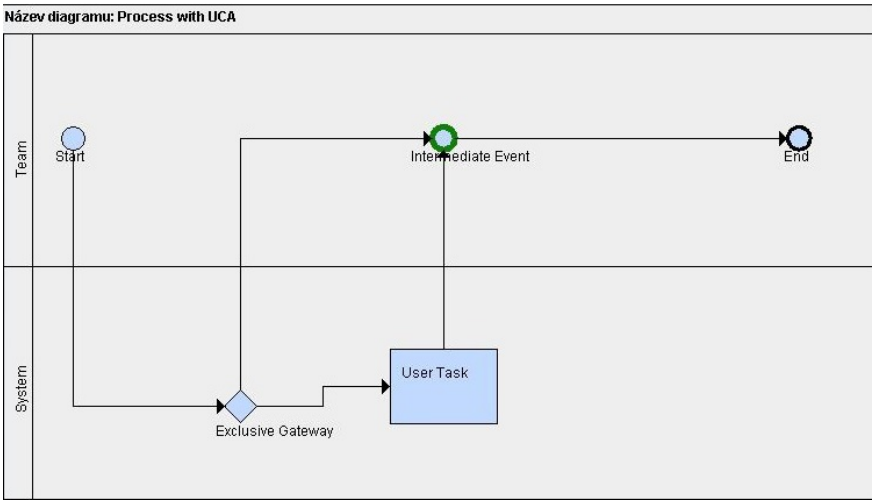
Skutečný výstup:



Obrázek 7.5: Testovací scénář 2 – nalezení undercover agent v business process definition

Vykreslí se diagram s názvem „Process with UCA“ a označeným artefaktem „Intermediate Event“ zeleným rámečkem.

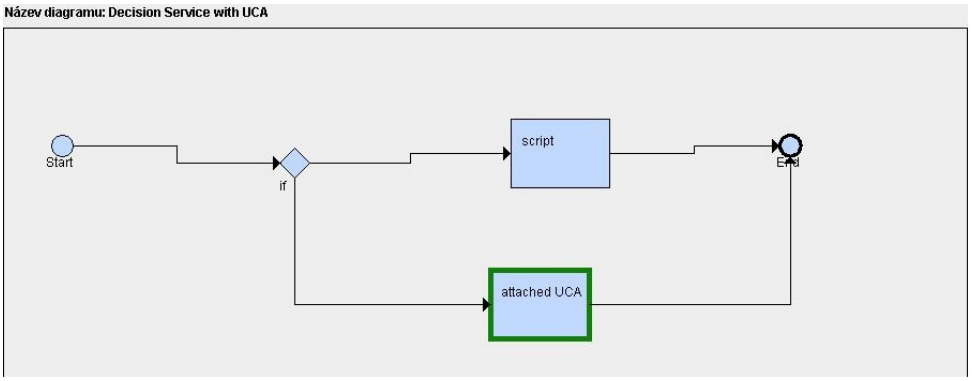
Skutečný výstup:



Obrázek 7.6: Testovací scénář 2 – nalezení undercover agent v process

Vykreslí se diagram s názvem „Decision Service with UCA“ a označeným artefaktem „attached UCA“ zeleným rámečkem.

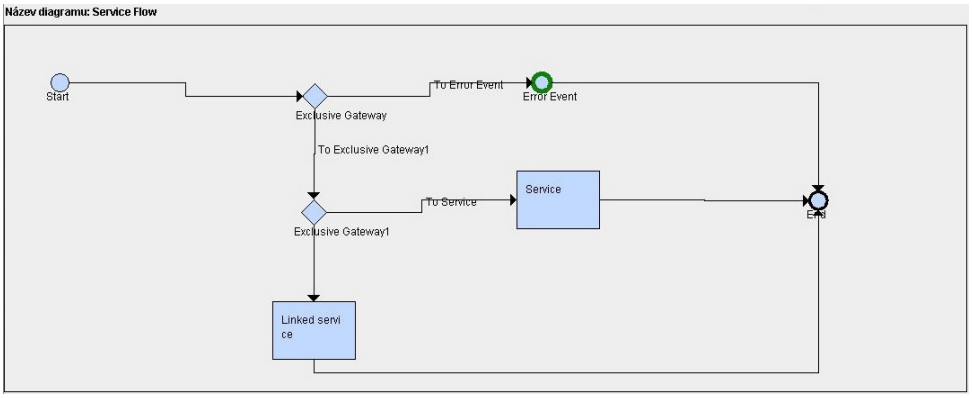
Skutečný výstup:



Obrázek 7.7: Testovací scénář 2 – nalezení undercover agent v service

Vykreslí se diagram s názvem „Service Flow“ a označeným artefaktem „Error Event“ zeleným rámečkem.

Skutečný výstup:



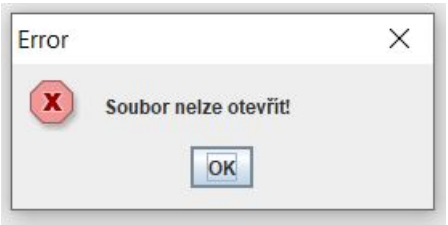
Obrázek 7.8: Testovací scénář 2 – nalezení undercover agent v service flow

4. Naimportujte přiložený scenar2.zip.

   Očekávaný výstup:

   Error s hláškou „Soubor nelze otevřít!“.

   Skutečný výstup:



Obrázek 7.9: Testovací scénář 2 – import ZIP souboru

5. Stiskněte OK a ukončete aplikaci křížkem v pravém horním rohu.

7.3.3 Testovací scénář 3

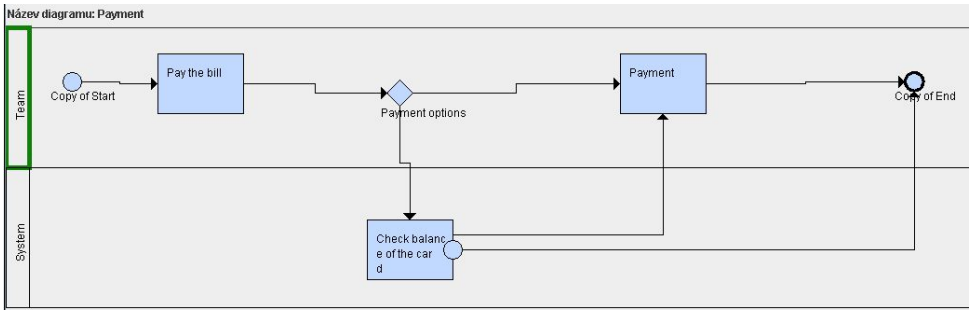
Vyhledání Team.

1. Spusťte aplikaci.
2. Naimportujte přiložený scenar3.twx.
3. Vyhledejte Team s názvem „MyTeam“.

   Očekávaný výstup – čtyři diagramy:

   Vykreslí se diagram s názvem „Payment“ a označeným týmem „Team“ zeleným rámečkem.

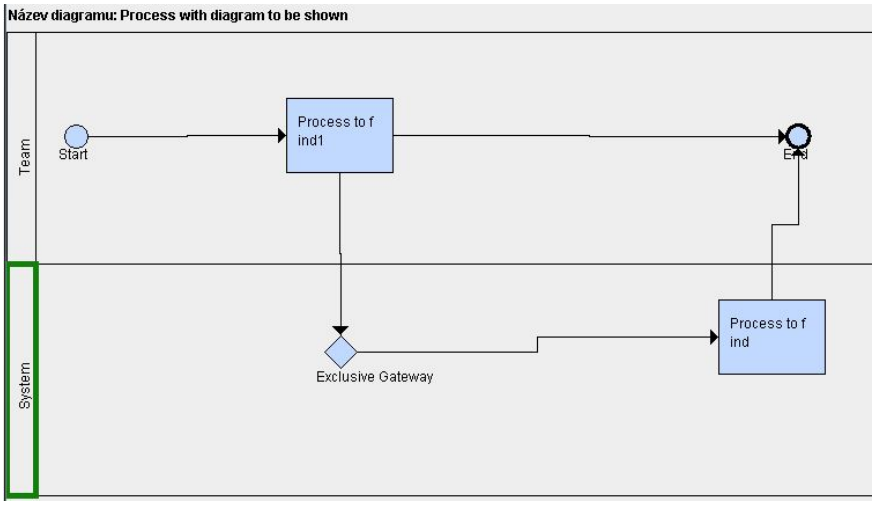
   Skutečný výstup:



Obrázek 7.10: Testovací scénář 3 – nalezení team v business process definition

Vykreslí se diagram s názvem „Process with diagram to be shown“ a označeným týmem „System“ zeleným rámečkem.

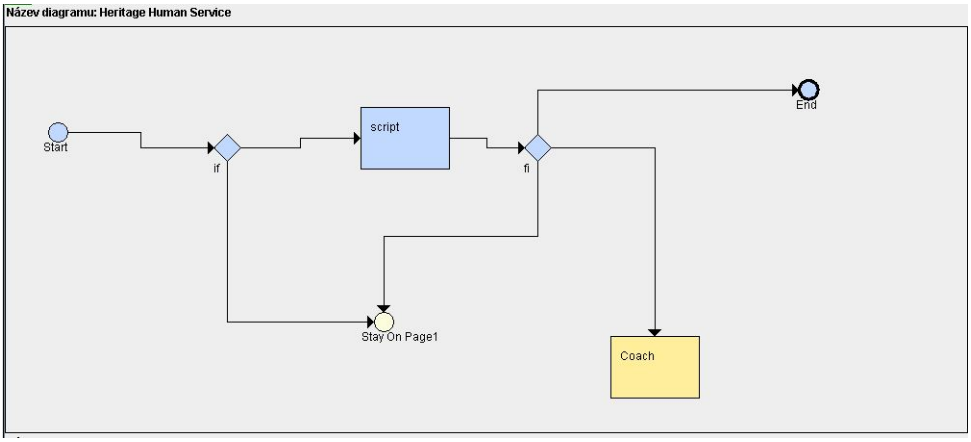
Skutečný výstup:



Obrázek 7.11: Testovací scénář 3 – nalezení team v process

Vykreslí se diagram s názvem „Heritage Human Service“.

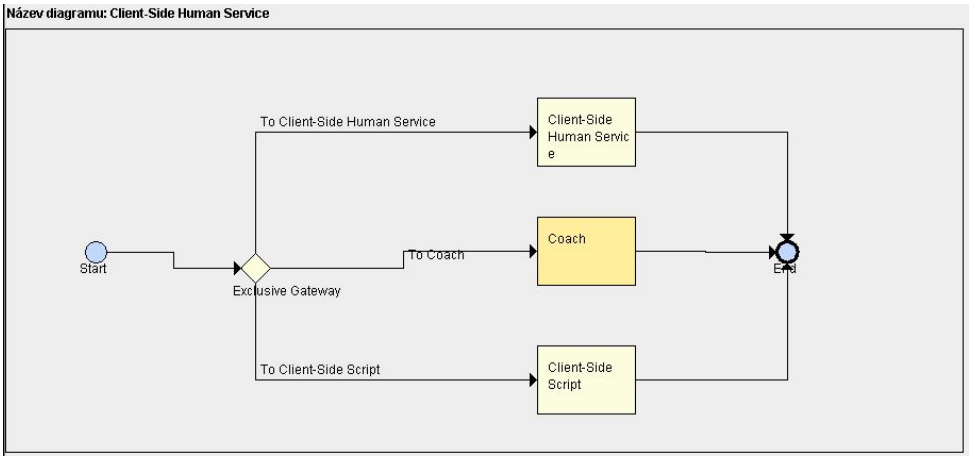
Skutečný výstup:



Obrázek 7.12: Testovací scénář 3 – nalezení team v heritage human service

Vykreslí se diagram s názvem „Client-Side Human Service“.

Skutečný výstup:



Obrázek 7.13: Testovací scénář 3 – nalezení team v client-side human service

4. Vyhledejte Process s názvem „Process with diagram to be shown“.

   Očekávaný výstup:

   Vypíše se „Artefakt není nikde použit.“

   Skutečný výstup:

Obrázek 7.14: Testovací scénář 3 – process není nikde použit

5. Ukončete aplikaci křížkem v pravém horním rohu.

7.3.4 Testovací scénář 4

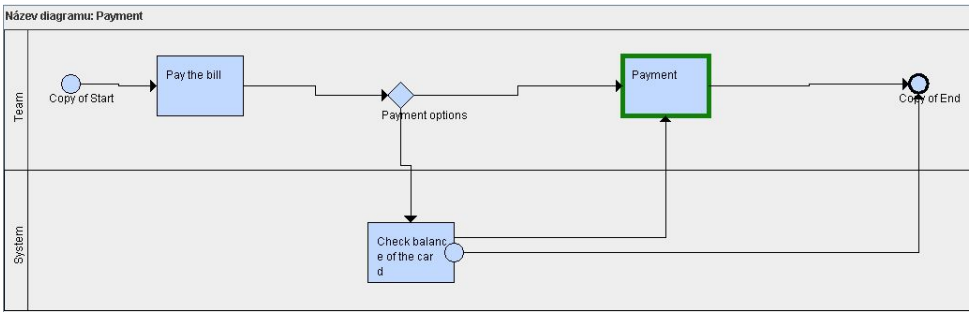
Vyhledání Heritage Human Service a Client-Side Human Service.

1. Spusťte aplikaci.
2. Naimportujte přiložený scenar4.twx.
3. Vyhledejte Heritage Human Service s názvem „HHS“.

   Očekávaný výstup – tři diagramy:

   Vykreslí se diagram s názvem „Payment“ a označeným artefaktem „Payment“ zeleným rámečkem.

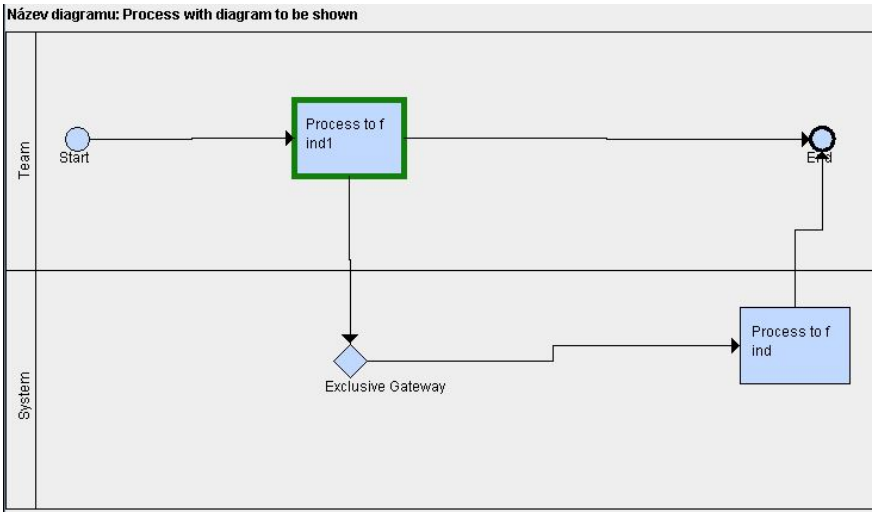
   Skutečný výstup:



Obrázek 7.15: Testovací scénář 4 – nalezení heritage human service v business process definition

Vykreslí se diagram s názvem „Process with diagram to be shown“ a označeným artefaktem „Process to find1“ zeleným rámečkem.

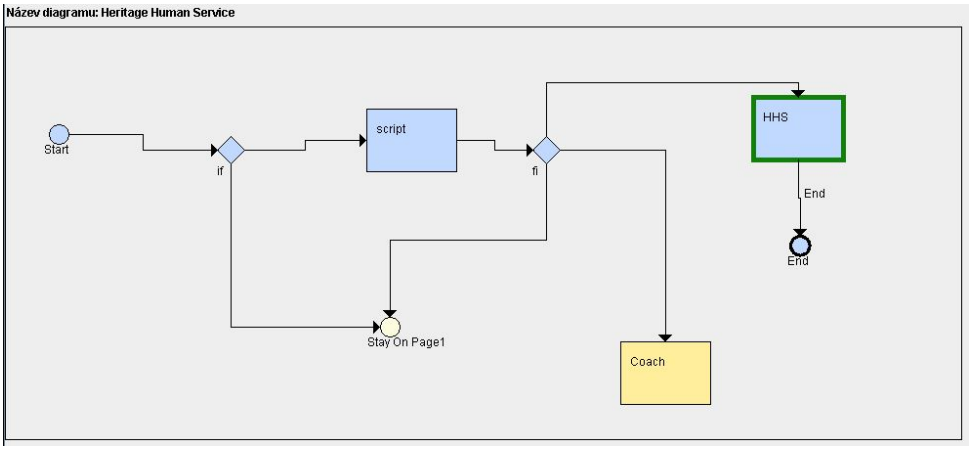
Skutečný výstup:



Obrázek 7.16: Testovací scénář 4 – nalezení heritage human service v process

Vykreslí se diagram s názvem „Heritage Human Service“ a označeným artefaktem „HHS“ zeleným rámečkem.

Skutečný výstup:



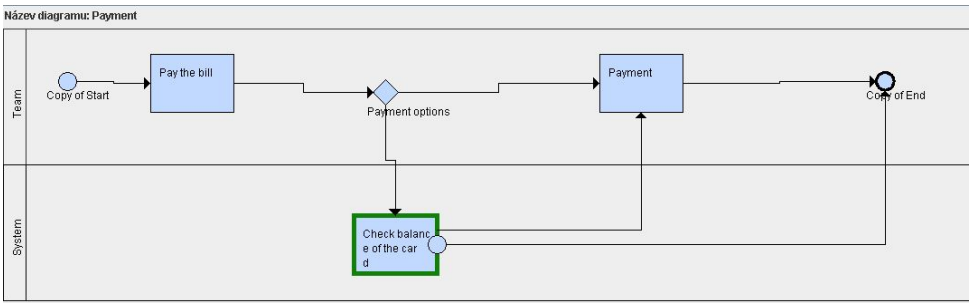
Obrázek 7.17: Testovací scénář 4 – nalezení heritage human service v jiné heritage human service

4. Vyhledejte Client-Side Human Service s názvem „Client-Side Human Service“.

   Očekávaný výstup – dva diagramy:

   Vykreslí se diagram s názvem „Payment“ a označeným artefaktem „Check balance of the card“ zeleným rámečkem.

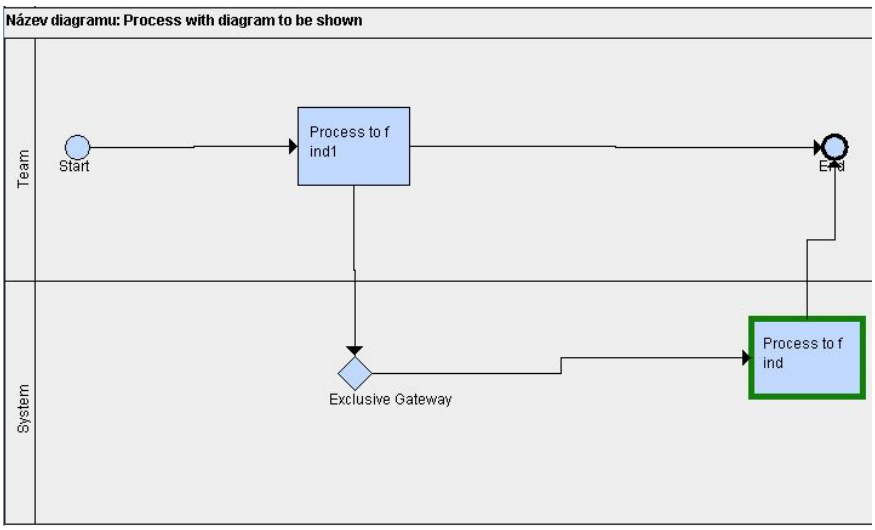
   Skutečný výstup:



Obrázek 7.18: Testovací scénář 4 – nalezení client-side human service v business process definition

   Vykreslí se diagram s názvem „Process with diagram to be shown“ a označeným artefaktem „Process to find“ zeleným rámečkem.

Skutečný výstup:



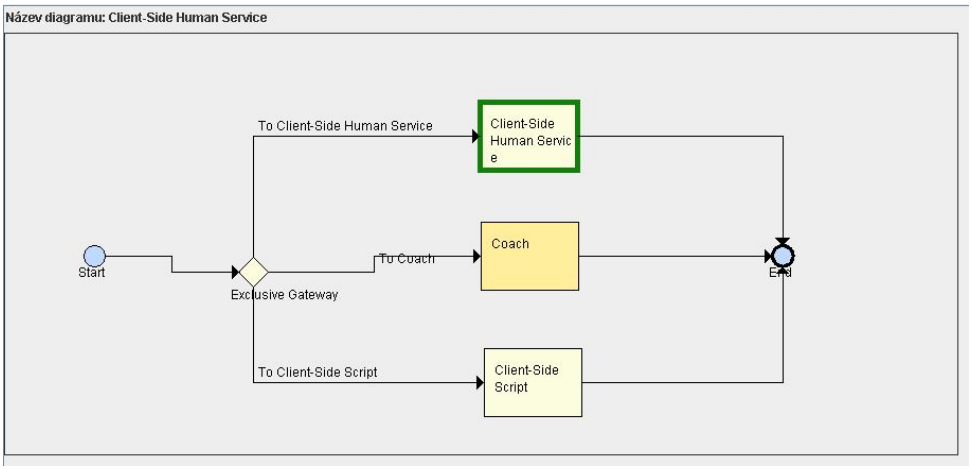
Obrázek 7.19: Testovací scénář 4 – nalezení cilent-side human service v process

5. Vyhledejte Client-Side Human Service s názvem „CSHS to find“.

   Očekávaný výstup – jeden diagram:

   Vykreslí se diagram s názvem „Client-Side Human Service“ a označeným artefaktem „Client-Side Human Service“ zeleným rámečkem.

   Skutečný výstup:



Obrázek 7.20: Testovací scénář 4 – nalezení client-side human service v jiné client-side human service

6. Ukončete aplikaci křížkem v pravém horním rohu.

7.3.5 Testovací scénář 5

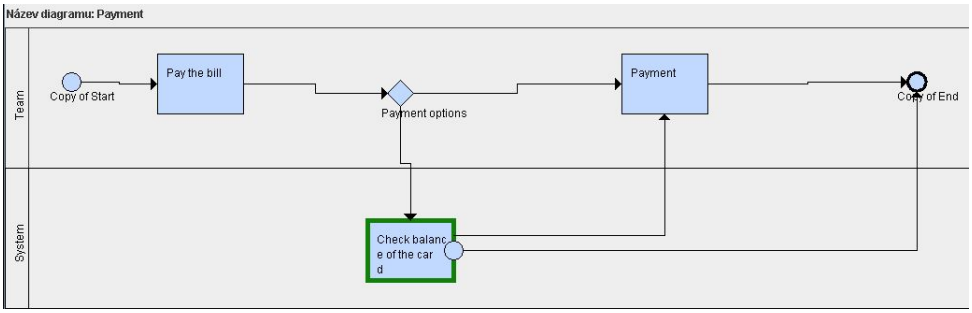
Vyhledání Service a Service Flow.

1. Spusťte aplikaci.
2. Naimportujte přiložený scenar5.twx.
3. Vyhledejte Service s názvem „Decision Service“.

   Očekávaný výstup – šest diagramů:

   Vykreslí se diagram s názvem „Payment“ a označeným artefaktem „Check balance of the card“ zeleným rámečkem.

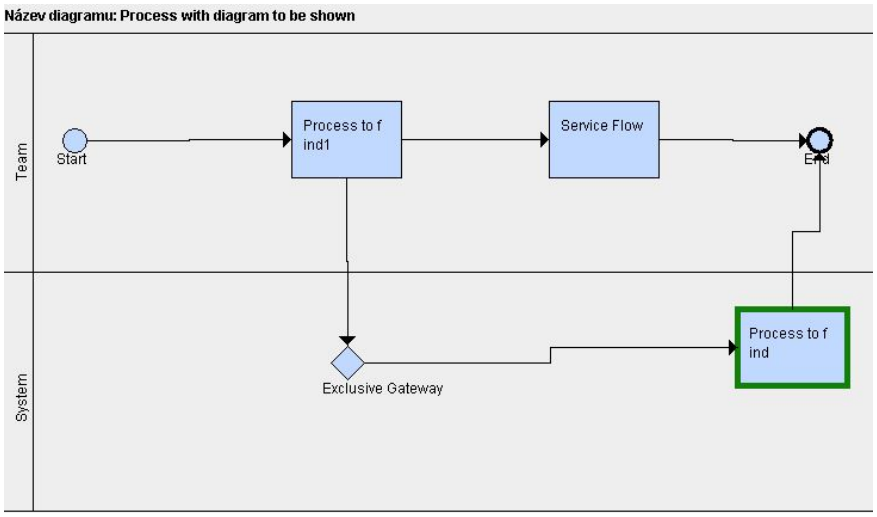
   Skutečný výstup:



Obrázek 7.21: Testovací scénář 5 – nalezení service v business process definition

Vykreslí se diagram s názvem „Process with diagram to be shown“ a označeným artefaktem „Process to find“ zeleným rámečkem.

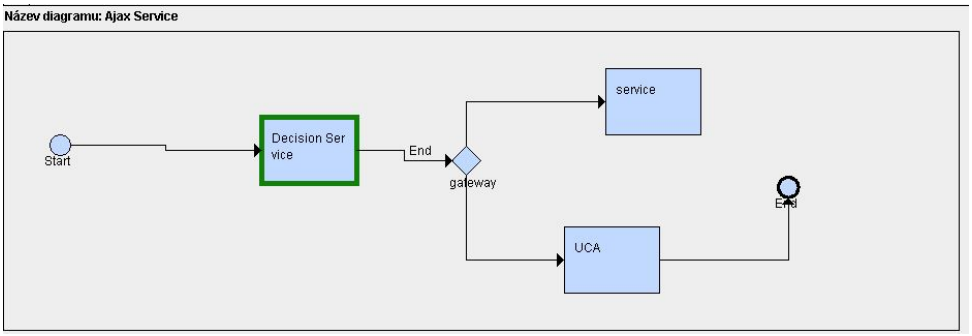
Skutečný výstup:



Obrázek 7.22: Testovací scénář 5 – nalezení service v process

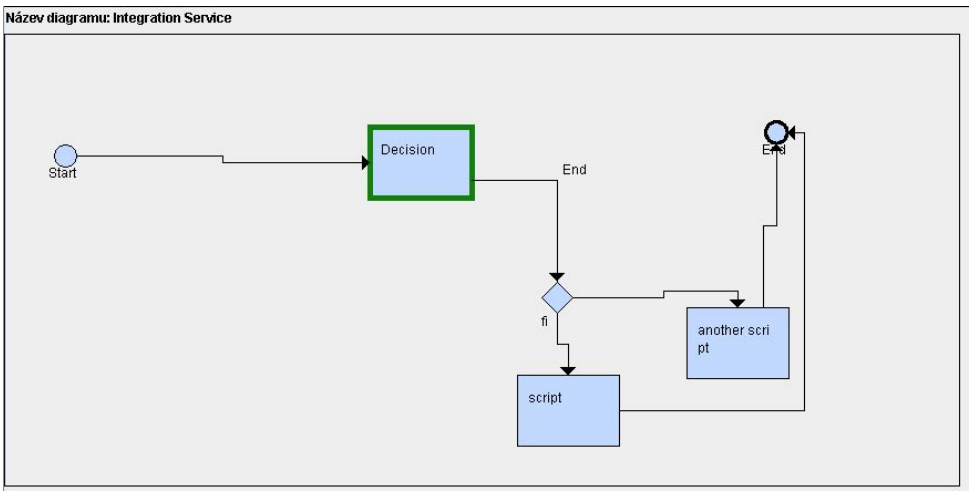
Vykreslí se diagram s názvem „Ajax Service“ a označeným artefaktem „Decision Service“ zeleným rámečkem.

Skutečný výstup:



Obrázek 7.23: Testovací scénář 5 – nalezení service v jiné service

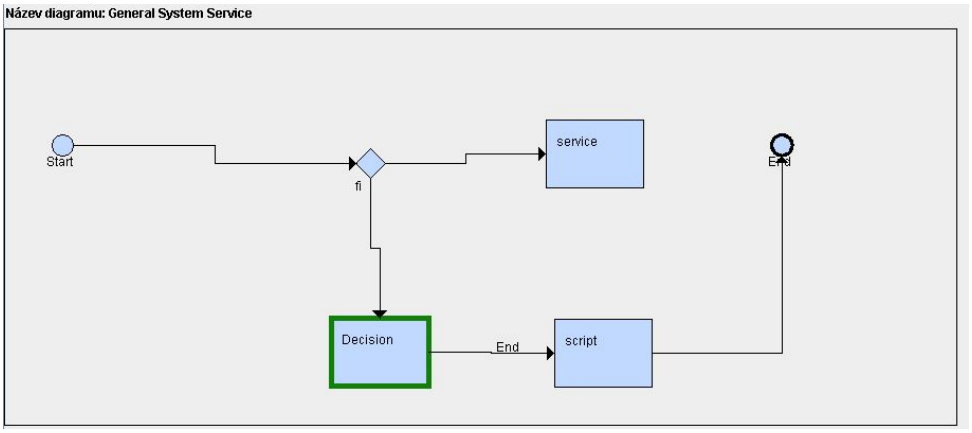
Vykreslí se diagram s názvem „Integration Service“ a označeným artefaktem „Decision“ zeleným rámečkem.



Obrázek 7.24: Testovací scénář 5 – nalezení service v integration service

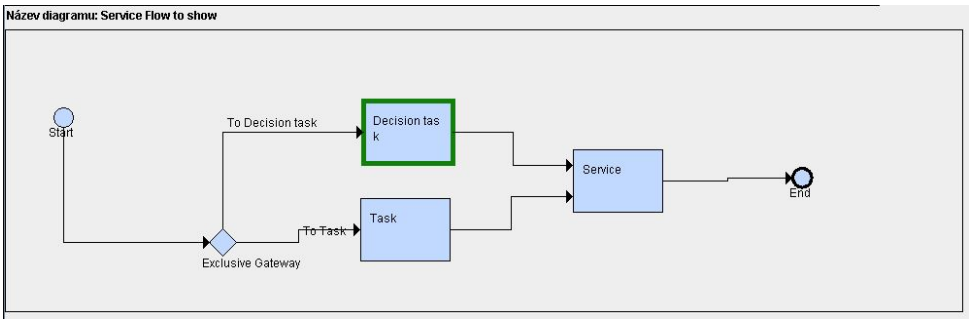
Vykreslí se diagram s názvem „General System Service“ a označeným artefaktem „Decision“ zeleným rámečkem.

Skutečný výstup:



Obrázek 7.25: Testovací scénář 5 – nalezení service v general system service

Vykreslí se diagram s názvem „Service Flow to show“ a označeným artefaktem „Decision task“ zeleným rámečkem.



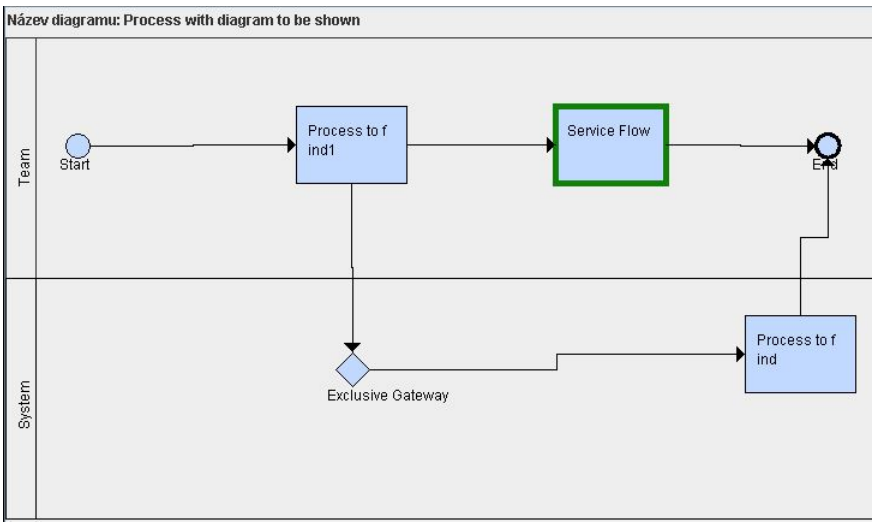
Obrázek 7.26: Testovací scénář 5 – nalezení service v service flow

4. Vyhledejte Service Flow s názvem „Service Flow to find“.

   Očekávaný výstup – dva diagramy:

   Vykreslí se diagram s názvem „Process with diagram to be shown“ a označeným artefaktem „Service Flow“ zeleným rámečkem.

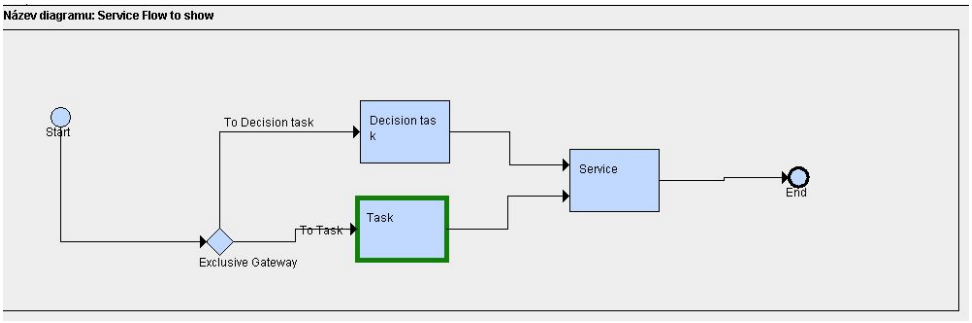
Skutečný výstup:



Obrázek 7.27: Testovací scénář 5 – nalezení service flow v process

Vykreslí se diagram s názvem „Service Flow to show“ a označeným artefaktem „Task“ zeleným rámečkem.

Skutečný výstup:



Obrázek 7.28: Testovací scénář 5 – nalezení service flow v jiné service flow

5. Ukončete aplikaci křížkem v pravém horním rohu.

7.3.6 Testovací scénář 6

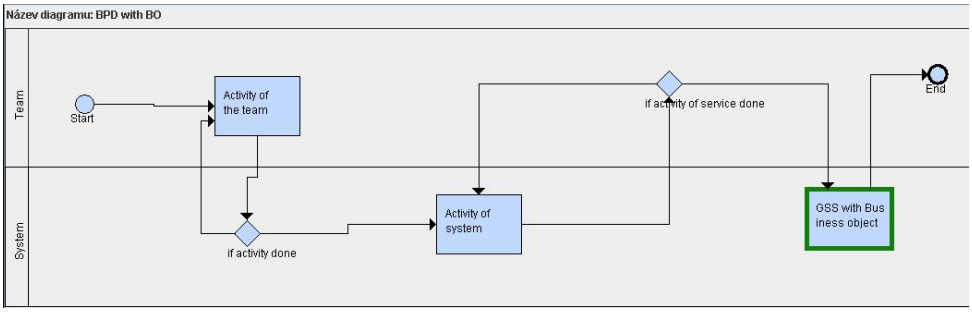
Vyhledávání Business Object.

1. Spusťte aplikaci.
2. Naimportujte přiložený scenar6.twx.
3. Vyhledejte Business Object s názvem „BusinessObjectToFind“.

Očekávaný výstup – sedm diagramů:

Vykreslí se diagram s názvem „BPD with BO“ a označeným artefaktem „GSS with Business object“ zeleným rámečkem.

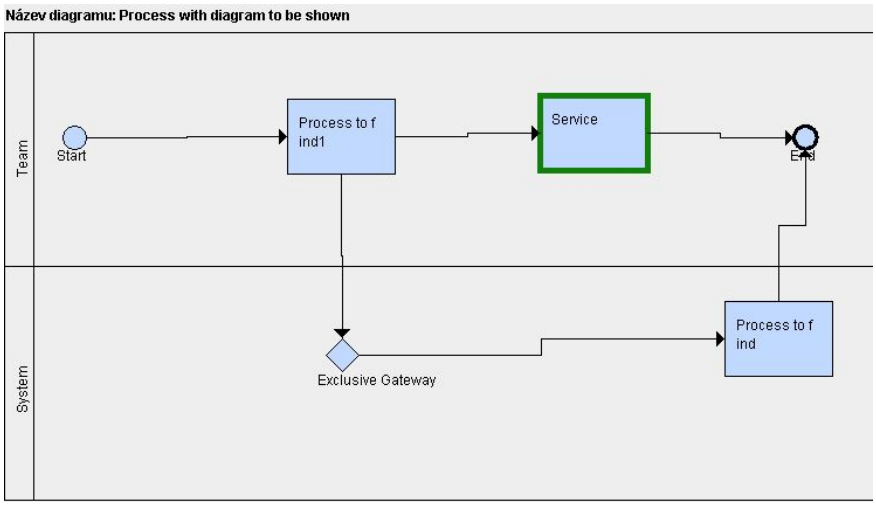
Skutečný výstup:



Obrázek 7.29: Testovací scénář 6 – nalezení business object v business process definition

Vykreslí se diagram s názvem „Process with diagram to be shown“ a označeným artefaktem „Service“ zeleným rámečkem.

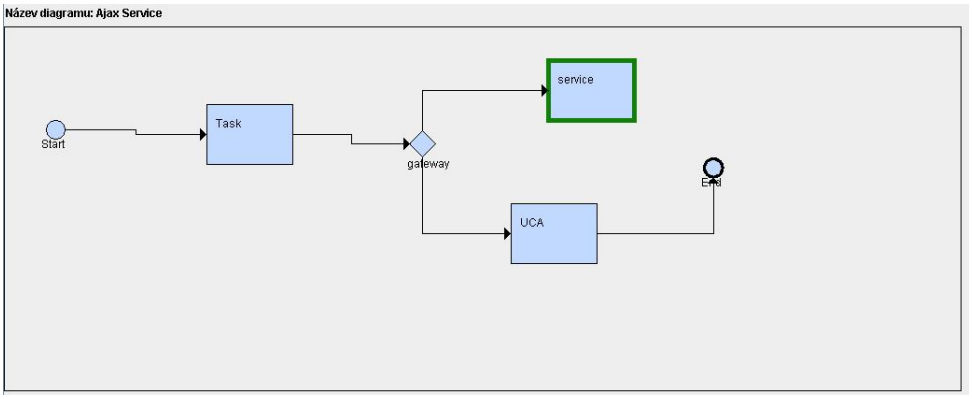
Skutečný výstup:



Obrázek 7.30: Testovací scénář 6 – nalezení business object v process

Vykreslí se diagram s názvem „Ajax Service“ a označeným artefaktem „service“ zeleným rámečkem.

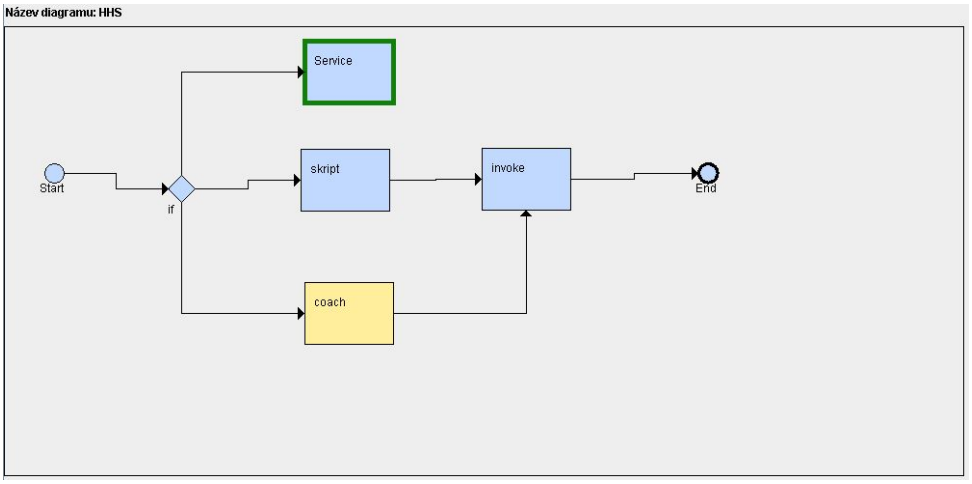
Skutečný výstup:



Obrázek 7.31: Testovací scénář 6 – nalezení business object v service

Vykreslí se diagram s názvem „HHS“ a označeným artefaktem „Service“ zeleným rámečkem.

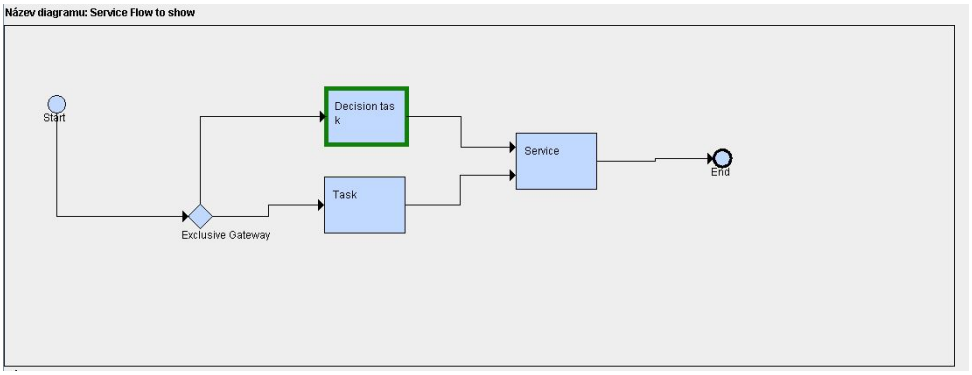
Skutečný výstup:



Obrázek 7.32: Testovací scénář 6 – nalezení business object v heritage human service

Vykreslí se diagram s názvem „Service Flow to Show“ a označeným artefaktem „Decision task“ zeleným rámečkem.

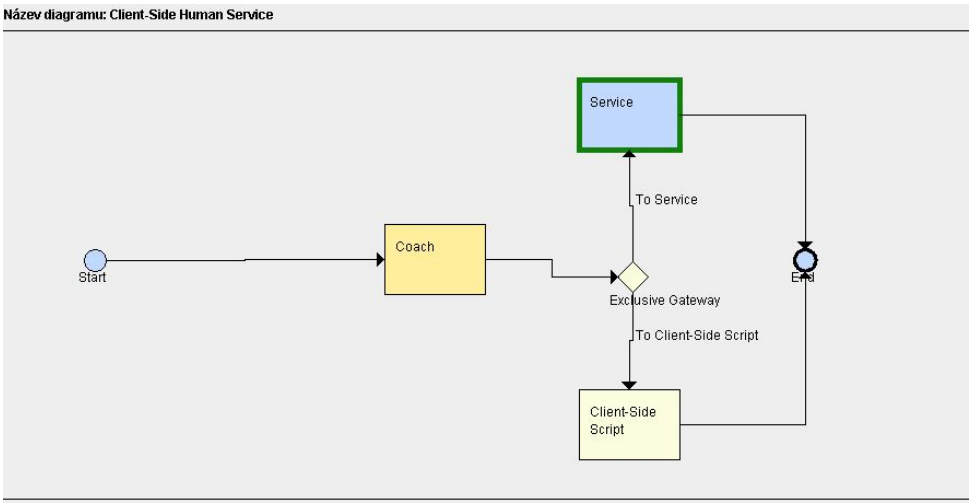
Skutečný výstup:



Obrázek 7.33: Testovací scénář 6 – nalezení business object v service flow

Vykreslí se diagram s názvem „Client-Side Human Service“ a označeným artefaktem „Service“ zeleným rámečkem.

Skutečný výstup:



Obrázek 7.34: Testovací scénář 6 – nalezení business object v client-side human service

Vypíše se název diagramu „BusinessObjectToShow“ bez vykreslení diagramu.

Skutečný výstup:

Obrázek 7.35: Testovací scénář 6 – nalezení business object v jiném business object

4. Ukončete aplikaci křížkem v pravém horním rohu.

7.4 Integrační testy

Na vytvořené testovací scénáře jsem napsala integrační testy aplikace. Integrační testy procházejí a testují, že se po průchodu aplikací najde očekávaný výstup. Tedy vytvořila jsem balíček IntegrationTest mezi ostatními balíčky na testování a v něm testovací třídy scenario1 až scenario6, které dělají stejné průchody jako testovací scénáře se stejným číslem.

Integrační testy nezobrazují grafický výstup, vše je testováno z dat uložených ve třídách.

7.5 Zhodnocení testování

Pomocí Unit testů byly testovány základní funkcionality aplikace bez nutnosti průchodu celou aplikací.

Integrační testy jsou automatizovaným testováním průchodů aplikací podle testovacích scénářů bez nutnosti ručního procházení aplikace.

Ověření funkčnosti podle testovacích scénářů na mém osobním laptopu proběhlo úspěšně. U prvního scénáře lze pod obrázky skutečných výstupu nalézt také obrázky diagramů vytvořených v process designeru. Při porovnání je viditelné, že diagramy jsou si podobné.

Testování pomocí testovacích scénářů dopadlo úspěšně také u čtyř lidí z Centra znalostního managementu. Odborný konzultant mi poskytl snapshot se složitějšími diagramy na vykreslení, jenž nebyl součástí testů, ale posloužil pro návrh další implementace aplikace. Možnosti ke zlepšení aplikace jsou k nalezení v následující kapitole 8.

Kapitola 8

Další rozšíření implementace a vylepšení aplikace

Z testovacích scénářů a také z otestování na snapshotu s těžšími diagramy na vykreslení vyplynuly další rozšíření implementace a vylepšení aplikace. Nejprve vypíšu vylepšení aplikace, které přinesly testovací scénáře, těmi jsou:

- Přidat informaci o průběhu importu snapshotu, nebo vyskakovací okénko s informací o dokončení importu snapshotu.
- Lépe popsat chybovou hlášku, která se zobrazí v případě, že uživatel zkouší importovat soubor s jinou koncovkou než TWX.
- Nezobrazovat posuvník v případě, kdy se všechny nalezené diagramy vejdou do okna.
- Pokud je hledán Team a zobrazí se diagram Client-side Human Service nebo Heritage Human Service, označit celý diagram zeleným rámečkem. Bez tohoto označení je matoucí důvod vykreslení diagramu.
- Při nalezení Business Objectu v jiném Business Objectu vypsat informaci, že tento objekt nemá diagram. Jinak to také může být matoucí, že se jen vypíše název bez vykreslení diagramu.
- Správně vykreslit popisky čar. Popisky čar se převážně správně vykreslují u Service, ale nefunguje vykreslení pospisků v Process a BPD, kde se nevykreslí.
- Dodělat ikonky artefaktům ve vykresleném diagramu. Tím si budou diagramy v aplikaci více podobné s diagramy v designeru. Aplikace je připravena uložit si ikonky ve třídě pro artefakt. Kvůli špatnému vytváření spustitelného JAR souboru, jsem musela odebrat vykreslování ikonek u diagramů, kde jsem to dokázala zprovoznit.

Z otestování aplikace na snapshotu s těžšími diagramy vyplynuly tyto další rozšíření implementace:

- Rozšířit aplikaci, aby fungovala na komplexnější diagramy. V aplikaci nefunguje vykreslování error handleru a eventu, které odchytávají errory

přímo z aktivit. Odchytávání errorů přímo z aktivit funguje pouze u process a business process definition. Ostatní diagramy nejsou v aplikaci kompletní, což se promítlo i v testovacích scénářích. Například v přiloženém CD v souboru porovnaniDiagramu.pdf je k nalezení ve scénáři 5, vyhledání Service v Integration Service, kde obrázku z aplikace chybí výstup error z activity a tím pádem se nevytvořila čára jdoucí do End eventu.

- Rozšíření na funkčnost pro pokrytí všech možností použití artefaktů pomocí Process designeru. Aplikace je vytvořená jen pro specifikované případy užití odborným konzultantem.
- Pro velké diagramy je možné urychlit aplikaci pomocí cache. Po prvním nalezení použití artefaktu by aplikace ukládala tato data. A při dalším hledání tohoto artefaktu by četla uložená data místo parsování všech XML souborů s možnými výskyty.
- Dalším možným vývojem je zaměření se na nalezený artefakt. V případě více artefaktů, pak možnost mezi artefakty přeskakovat pomocí šipek. To ulehčí uživateli nalezení použití artefaktu ve velkém diagramu, který přesahuje obrazovku.

Kapitola 9

Závěr

Cílem bakalářské práce bylo analyzovat systém verzování procesních aplikací na platformě IBM BPM. Navrhnout a implementovat nástroj pro analýzu artefaktů, který uživateli zobrazí textovou i grafickou informaci o tom, v jaké části aplikace je daný artefakt použitý.

Do teoretické části patří rešerše systému na verzování procesních aplikací na platformě IBM BPM, analýza metod na parsování XML souborů a na vizualizaci. Nejprve jsem uvedla definice Business process a BPMN. Sepsala jsem historii z hlediska vývoje systému na verzování a popsala jsem nástroje IBM BPM a Process designeru. Také jsem vysvětlila, jak vytvořit a co je to procesní aplikace a toolkit. Popsala jsem, jak exportovat a importovat snapshoty z Process designeru. Dále jsem popsala adresářovou strukturu a artefakt. Nakonec jsem v teoretické části analyzovala a popsala metody řešení vyhledávání v XML souborech a vizualizace nalezných diagramů.

Návrhem implementace popisuji funkční a nefunkční požadavky aplikace. Popsala jsem případy užití a průchody aplikací. Také jsem vytvořila UML diagram tříd a popsala účel jednotlivých tříd v daném diagramu. Nakonec jsem vypsala použité technologie na vizualizaci nalezených diagramů a vyhledávání v XML souborech.

V implementační části jsem popsala implementaci parsování XML souborů, vyhledávání artefaktů v nich a vizualizaci diagramů, ve kterých je použit vyhledávaný artefakt.

Testování aplikace popisuje regresní testování, neboli Unit testy a integrační testy aplikace. Dále jsem popsala testovací scénáře, ukázala skutečné výsledky scénářů a napsala zhodnocení funkčnosti aplikace od lidí z CZM. Vyplněné testovací scénáře jsou k nalezení na CD v archivu vysledkyTestovani.zip.

Poslední částí jsem popsala další rozšíření implementace a vylepšení aplikace, které vzešly z testovacích scénářů a snapshotu s těžšími diagramy na vykreslování.

Nakonec jsem vytvořila uživatelskou příručku, kterou lze nalézt v příloze C „Uživatelská příručka“.

Řešení vyhledávání artefaktu v procesní aplikaci je první verzí, kterou lze dále rozvíjet. Rozšíření na funkčnost pro pokrytí všech možností použití artefaktů pomocí Process designeru je nad rámec jedné bakalářské práce. Ovšem cíle práce jsem splnila a aplikace by mohla být využitelná ve skupině

lidí, kteří používají Process designer a potřebují najít použití artefaktu v případech užití, které jsem zmínila v kapitole .2.

Literatura

[1] SVOZILOVÁ, Alena. *Zlepšování podnikových procesů.* Vydání 1. Praha: Grada, 2011. ISBN isbn978-80-247-3938-0.

[2] *techopedia* [online]. [cit. 29.12.2019]. Dostupné z: https://www.techopedia.com/definition/1168/business-process.

[3] *Object Management Group Business Process Model and Notation* [online]. ©Copyright 1997-2018 Object Management Group [cit. 1.12.2018]. Dostupné z: http://www.bpmn.org.

[4] *Object Management Group* [online]. ©Copyright 2020 Object Management Group [cit. 2.1.2020]. Dostupné z: https://www.omg.org/bpmn/Documents/Introduction_to_BPMN.pdf?fbclid=IwAR26q2GP0GJuqDcGDWipIolRQIp7TVzFbp4IsiqvISLnS3j70_6TS_phMVg.

[5] KOLBAN, Neil [online]. *Kolban's Book on IBM Business Process Management.* Prosinec 2014. Dostupné z: http://neilkolban.com/ibm/wp-content/uploads/2014/12/Kolbans-IBPM-Book-2014-12.pdf.

[6] *IBM Knowledge Center* [online]. ©Copyright IBM Corporation 1994, 2018 [cit. 29.12.2018]. Dostupné z: https://www.ibm.com/us-en/marketplace/ibm-business-automation-workflow.

[7] *IBM Knowledge Center* [online]. ©Copyright IBM Corporation 1994, 2018 [cit. 29.12.2018]. Dostupné z: https://www.ibm.com/support/knowledgecenter/cs/SSV2LR/com.ibm.wbpm.main.doc/topics/ibmbmp_overview.html.

[8] *ReviverSoft* [online]. ©Copyright 2019 Corel Corporation [cit. 10.1.2019]. Dostupné z: https://www.reviversoft.com/en/file-extensions/jar.

[9] *tutorialspoint* [online]. ©Copyright 2018 [cit. 26.12.2018]. Dostupné z: https://www.tutorialspoint.com/java_xml.

[10] *Studytonight* [online]. ©Copyright 2019 Stadytonight [cit. 3.1.2019]. Dostupné z: https://www.studytonight.com/java/java-awt.php.

[11] *javatpoint* [online]. ©Copyright 2011-2018 www.javatpoint.cz [cit. 3.1.2019]. Dostupné z: `https://www.javatpoint.com/java-swing`.

[12] *ITnetwork.cz* [online]. ©Copyright 2019 itnetwork.cz [cit. 3.1.2019]. Dostupné z: `https://www.itnetwork.cz/java/javafx/java-tutorial-uvod-do-javafx`.

[13] HRIADELOVÁ, Anna Mária. *Vizualizace změn v procesních aplikacích.* Praha: 2019. Dostupné z: `https://dspace.cvut.cz/handle/10467/82365`. Bakalářká práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů. Vedoucí práce Ing. Lukáš Zoubek.

[14] GRASSEOVÁ, Monika, Radek DUBEC a Roman HORÁK. *Procesní řízení ve veřejném sektoru: Teoretická východiska a praktické příklady.* Vydání 1. Brno: Computer Press, 2008. ISBN isbn978-80-251-1987-7.

[15] SILVER, Bruce. *BPMN method and style: with BPMN implementer's guide.* Second edition. Aptos: Cody-Cassidy Press, 2011. ISBN isbn978-09-823-6811-4.

Příloha A

Seznam použitých zkratek

- API – Application Programming Interface
- AWT – Abstract Windowing Toolkit
- BAW – Business Automation Workflow
- BPEL – Business Process Execution Language
- BPM – Business Process Management
- BPMN – Business Process Model and Notation
- DOM – Document Object Model
- GUI – Graphical User Interface
- IBPM – IBM BPM
- SAX – Simple API for XML
- SCA – Service Component Architecture
- StAX – Streaming API for XML
- WPS – WebSphere Process Server
- WLE – WebSphere Lombardi Edition
- XPath – XML Path Language
- XSLT – Extensible Stylesheet Language Transformations

Příloha B

Obsah přiloženého CD

- BachelorThesis.pdf – text práce v PDF souboru
- porovnaniDiagramu.pdf – diagramy nalezené aplikací z testovacích scénářů a diagramy z vytvořené v process designeru
- vysledkyTestovani.zip – vyplněné testovací scénáře lidmi z Centra znalostního managementu
- tex.zip – potřebné soubory a obrázky k vytvoření PDF souboru v texu
- twxSearch.jar – spustitelný soubor bez nutnosti kompilace zdrojových kódů, ovšem je nutné mít JRE
- twxSearch.zip – zdrojové kódy implementační části bakalářské práce

Příloha C

Uživatelská příručka

Aplikaci lze spustit několika způsoby. Je potřeba mít nainstalovaný JRE, nebo-li Java Runtime Environment, verze 1.8 a vyšší. Poté dvojklikem spustit twxSearch.jar. Při přístupu ke zdrojovým kódům, je možné spustit aplikaci pomocí vývojového prostředí. K tomu je nutné mít Java verze 1.8 a Apache Maven verze 3.6.0. Pak stačí spustit `build` a `run` ve vývojovém prostředí. Další možností je spuštění přes Apache Maven plugin. Při možnosti spuštění maven pluginu ve vývojovém prostředí je třeba spustit plugin `package`. Vytvoří se složka `target`, která obsahuje, mimo jiné, soubor `twxsearch-1.0-SNAPSHOT-jar-with-dependencies.jar`. Tento JAR soubor je možné spustit dvojklikem. Alternativně lze provést `build` v příkazové řádce pomocí příkazu `mvn package` ve složce twxsearch. Poté spustit aplikaci přechodem do složky target a zde zavolat příkaz
`java -jar twxsearch-1.0-SNAPSHOT-jar-with-dependencies`.

Po spuštění se otevře aplikace jako na obrázku C.1, jen bez barevných rámečků.

Obrázek C.1: twxSearch aplikace

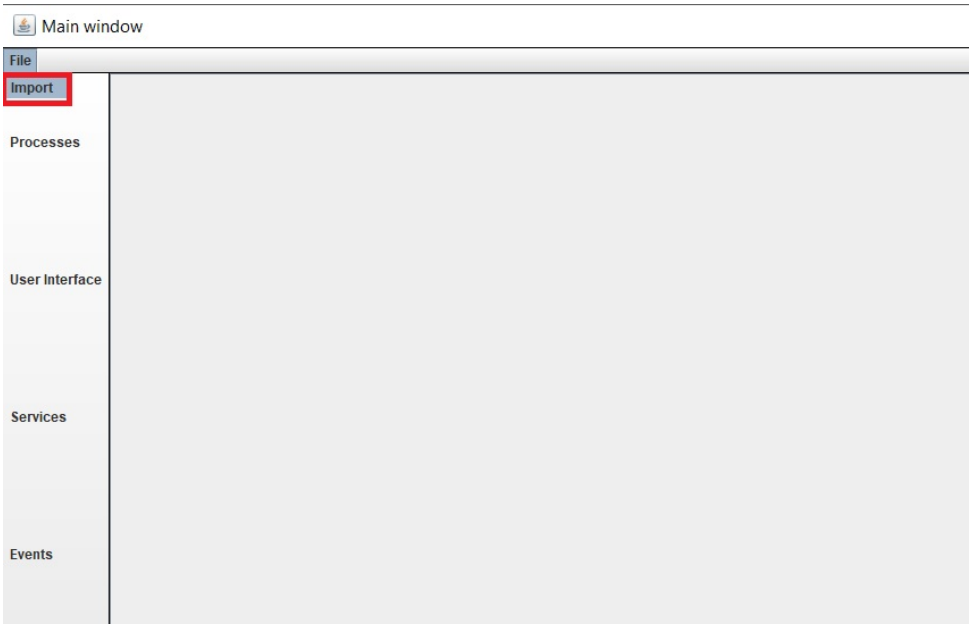
Aplikaci je možné vypnout pomocí křížku v pravém horním rohu, na obrázku C.1 označený červeným rámečkem.

Do aplikace lze importovat snapshot, import je k nalezení v levém horním rohu v menu File, na obrázku C.1 je zvýrazněn modrým rámečkem. Po rozkliknutí importu se objeví průzkumník souborů, pomocí kterého lze nalézt umístění snapshotu.

Po importu je možné vybrat artefakt, pro který má aplikace nalézt použití. Artefakty jsou roztříděné do skupin podle typu, každý typ má své vlastní menu. Jejich seznam je k nalezení v levé části aplikace, na obrázku C.1 je tato část zvýrazněná zeleným rámečkem.

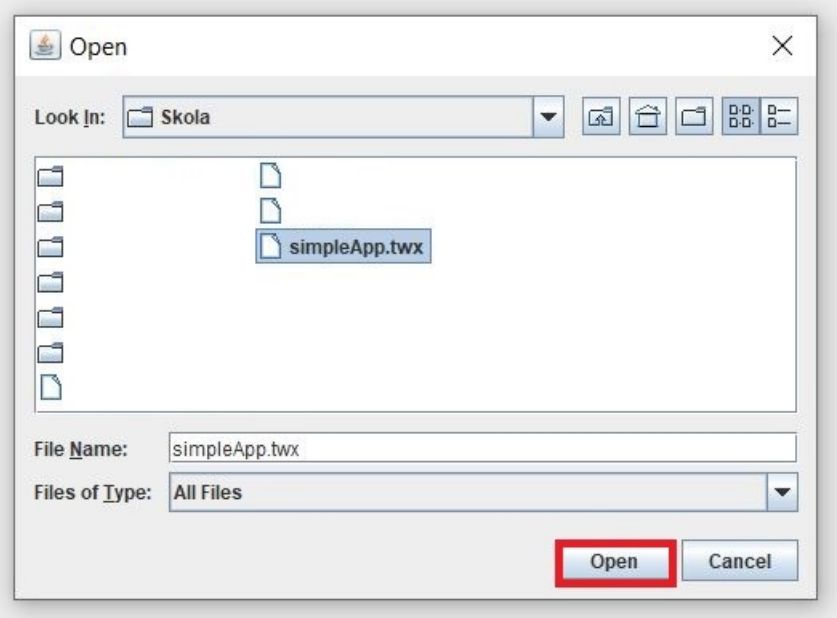
Jeden z možných průchodů aplikací je:

Spustit aplikaci pomocí twxSearch.jar. Otevře se okno aplikace. Stisknout import, jako je zvýrazněno na obrázku C.2 červeným rámečkem.



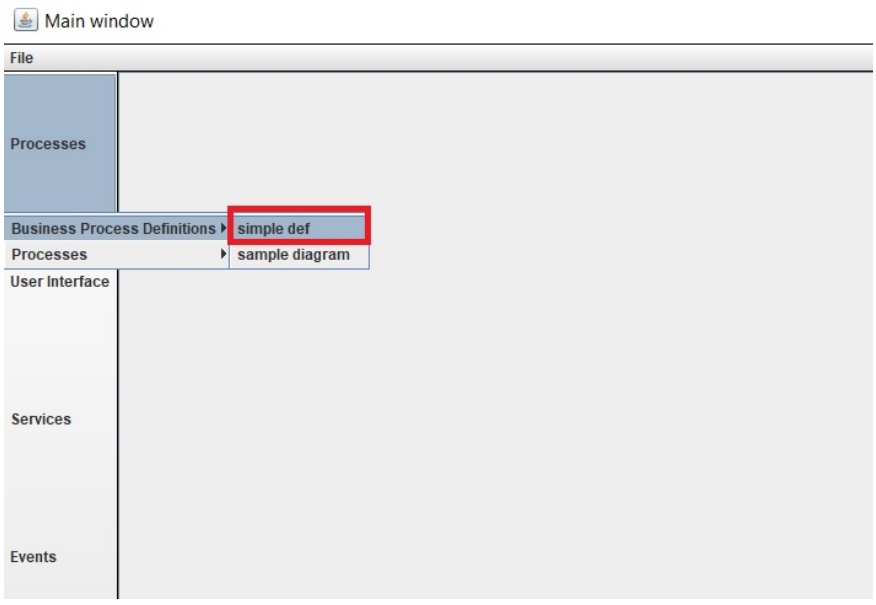
Obrázek C.2: Import snapshotu

Vybrat snapshot na import v průzkumníku souborů a stisknout Open, jako je zvýrazněno na obrázku C.3 červeným rámečkem.



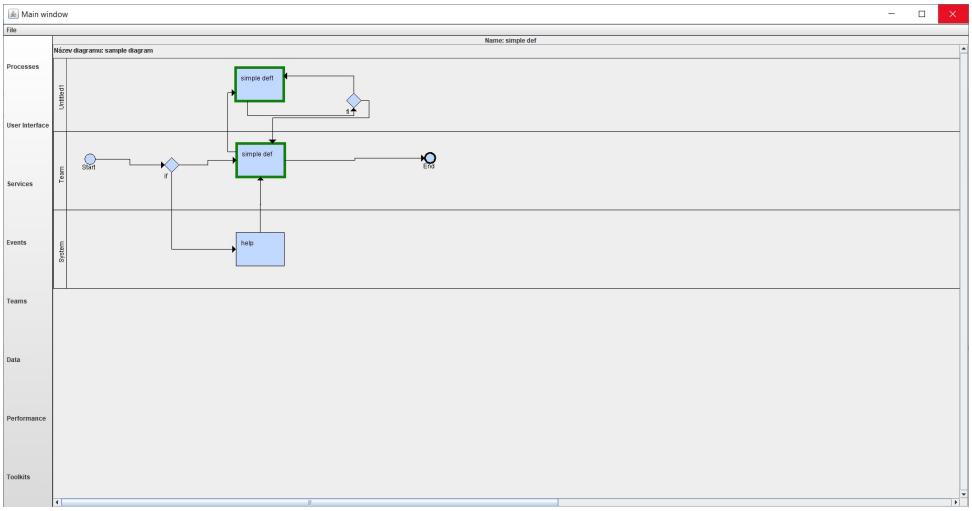
Obrázek C.3: Průzkumník souborů

Provede se import snapshotu a naplnění všech skupin menu v levé části aplikace. V tomto příkladě je vybrána business process definition s názvem „simple def“, výběr artefaktu je na obrázku C.4 zvýrazněn červeným rámečkem.



Obrázek C.4: Výběr business process definition

Po tomto kroku je viditelný diagram, ve kterém je hledaný artefakt použitý jako na obrázku C.5.



Obrázek C.5: Vykreslený diagram použití

Aplikaci je ukončena křížkem v pravém horním rohu, který je na obrázku C.5 v červeném vyplněném obdélníčku.