TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Plugin pro tvorbu sekvenčních diagramů v programu EXAM

Bakalářská práce

Studijní program: B0714A270001 Mechatronika
Autor práce: **Aleš Vykouk**
Vedoucí práce: Ing. Roman Špánek, Ph.D.
Ústav mechatroniky a technické informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Plugin pro tvorbu sekvenčních diagramů v programu EXAM

Jméno a příjmení: **Aleš Vykouk**
Osobní číslo: M19000111
Studijní program: B0714A270001 Mechatronika
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2021/2022**

Zásady pro vypracování:

1. Seznamte se s tvorbou pluginu pro program EXAM, který umožní dokončení kódu v jazyce Python pro sekvenční diagramy programu EXAM.
2. Vytvořte datovou strukturu, která bude obsahovat a ukládat návrhy pro dokončení kódu v sekvenčním diagramu.
3. Navrhněte systém pro dokončování kódu, který vezme v potaz všechny existující datové typy, bude navrhovat pouze typy, které mají v daném bodě smysl, a navrhované prvky budou uloženy v relační databázi.
4. Plugin musí být realizován pro prostředí Eclipse (program EXAM jej využívá). Připravte UI pro realizovaný plugin.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30–40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] PECINOVSKÝ, R, Python kompletní příručka jazyka pro verzi 3.9, Praha : Grada Publishing, 2020; 2020

[2] DATE, C. J. Database design and relational theory, Healdsburg, California, USA : Apress, [2019]

Vedoucí práce:	Ing. Roman Špánek, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	12. října 2021
Předpokládaný termín odevzdání:	16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2021

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

13. května 2022 Aleš Vykouk

Poděkování

Chtěl bych poděkovat vedoucímu práce panu Ing. Romanu Špánkovi, Ph.D. za přínosné rady a poznámky při tvorbě práce. Dále bych chtěl poděkovat firmě MicroNova AG za možnost zpracovat toto téma jako bakalářskou práci. V neposlední řadě bych chtěl poděkovat své rodině a přítelkyni za podporu při studiu a tvorbě této práce.

Abstrakt

Práce se zabývá sestavením softwarového řešení jako proof of concept (POC) pro ověření využitelnosti data mining algoritmů apriori a tvorby asociačních pravidel jako základu pro navrhování relevantních elementů při tvorbě sekvenčních diagramů v softwaru EXAM. Jelikož žádný takovýto systém není v současné době v aplikaci EXAM implementován, je třeba ověřit daný přístup k řešení problému.

Práce popisuje implementaci navrhování elementů do sekvenčních diagramů založeného na obsahu již existujících diagramů v určitém EXAM modelu, navrhuje způsob získání potřebných dat z modelu a způsob jejich zpracování pro vytvoření pravidel, která slouží jako základ pro napovídání. Dále je v práci řešena tvorba pluginu integrovaného do aplikace EXAM v prostředí Eclipse, který vytvořená pravidla ukládá do relační databáze a je schopen v této databázi patřičně vyhledávat. Plugin je schopen navrhnout prostřednictvím uživatelského rozhraní relevantní elementy pro tvorbu konkrétního sekvenčního diagramu, které lze jednoduše použít v jakékoliv části právě vyvíjeného diagramu.

V textu práce je popsáno řešení výše zmíněných cílů. Práce samotná je členěna na dvě hlavní části. V první části byly popsány termíny a témata potřebná pro pochopení motivace a zpracování zadání, konkrétně rozšiřovaná aplikace EXAM, platforma Eclipse a použité programovací jazyky. Je zde také představeno téma data mining. Druhá stěžejní část práce se zaměřuje na vývoj softwarového řešení, konkrétně na získání základních dat pomocí Groovy skriptu, zpracování těchto dat pomocí Python skriptu a nakonec využití vytvořených pravidel v Eclipse pluginu uvnitř softwaru EXAM.

V neposlední řadě je v práci popsán přínos zpracování zadání jako ověření konceptu a porovnání určitých vlastností navrženého řešení s potenciálním využitím zpracovaného přístupu v produkčním nasazení programu.

Klíčová slova

Apriori, asociační pravidla, Eclipse, EXAM, plugin, sekvenční diagram

Abstract

This thesis deals with the implementation of a software solution as a proof of concept (POC) regarding the usage of data mining algorithms, namely Apriori and association rule mining, as the basis for suggestion of relevant elements in development of a sequence diagram in EXAM software. Since no such system is currently implemented in EXAM, it is needed to verify the given approach to the problem.

The thesis describes the implementation of element suggestion for sequence diagram creation based on the content of already existing diagrams in a given EXAM model. It proposes a way of obtaining required data from the EXAM model and a subsequent process of using this data for creating rules which are then used for element suggestion. Furthermore, it deals with the creation of an Eclipse plugin integrated into EXAM application that saves the created rules into a relational database and is able to accordingly search through it. Through UI the plugin is able to suggest relevant elements that can be used simply and in whatever part of a particular sequence diagram.

The text of this work describes the solution to the aforementioned goals. The text itself is divided into two main parts. Terms and topics that are needed to understand the motivation and solution of the given task are described in the first segment, which include the expanded application EXAM, Eclipse platform, used programming languages and an introduction to data mining. The second key part focuses on development of the software solution, namely acquiring base data from an EXAM model through a Groovy script, processing this data using a Python script and finally using the created rules inside the Eclipse plugin for EXAM.

Last but not least, the work describes benefits of the solution as verification of the concept as well as a comparison between certain features of the proposed solution and a potential use of the given approach in a production deployment of the program.

Keywords

Apriori, association rules, Eclipse, EXAM, plugin, sequence diagram

Obsah

Úvod

Zvyšující se složitost a bezpečnost novodobých automobilů se ruku v ruce pojí s hojným využitím elektronických řídících systémů. Automobilky na tyto systémy spoléhají čím dál více a jsou na ně přirozeně kladeny vysoké nároky týkající se jejich spolehlivosti.

Většina moderních ovládacích systémů motorů je digitální. Typický řídící systém pak využívá mikroprocesor, který lze chápat jako počítač se speciálním účelem. Elektronické řízení motoru se vyvinulo z palivového řídícího systému, který využíval diskrétní analogové komponenty k vysoce přesnému ovládání paliva a zapalování díky 32bitovým mikroprocesorům integrujícím digitální řízení hnacího ústrojí. Motivací pro vývoj složitějších digitálních řídících systémů byly přísné regulace ohledně výfukových plynů a spotřeby vozů. [1]

Kýženou spolehlivost řídících systémů je tedy kvůli jejich zvyšující se složitosti třeba důkladně testovat. A právě touto oblastí, mimo jiných, se zabývá firma MicroNova AG, ve spolupráci s níž bylo téma práce vybráno a realizováno.

Motivace pro zpracování tématu vyplývá ze zaměření softwaru EXAM, který vyvíjí MicroNova AG, na automatické testování elektronických systémů v automobilovém průmyslu. Tento software využívají například „testeři“, kteří nutně nemusí mít zkušenosti s programováním ve vyšších programovacích jazycích. Kvalita služby, která se poskytuje prostřednictvím softwaru EXAM, se mimo jiné měří skrze uživatelskou přívětivost a jednoduchost při vytváření již zmíněných testů. A právě na to se práce zaměřuje. Tedy udělat proces vytváření testovacích sekvencí více přímočarý, uživatelsky přívětivý a časově účinný skrze automatickou nápovědu entit při popisování jednotlivých testů a operací v nich.

Předmětem zvoleného tématu je vytvoření pluginu pro software EXAM v prostředí Eclipse a jazyce Java jako proof of concept (ověření konceptu). Tento plugin by měl obstarat uživateli adekvátní automatickou nápovědu, založenou na asociační predikci entit, při tvorbě sekvenčních diagramů, které jsou dále interpretovány v jazyce Python. Vytvořené sekvenční diagramy tvoří testovací případy a sekvence, které jsou poté využité například v HIL (hardware in the loop) nebo SIL (software in the loop) testování.

Před začátkem vlastního vývoje pluginu bylo třeba se seznámit s programem EXAM, prostředím Eclipse, programovacími jazyky Java, Groovy a Python. Bylo také třeba se seznámit se základy data mining a jeho využití pro zadané téma.

1 Teoretický rozbor

1.1 Extended automation method

Akronym EXAM popisuje sousloví extended automation method (rozšířená automatizační metoda). Tento software je vyvíjený firmou MicroNova AG na platformě Eclipse. Jedná se o Freeware, nejedná se však o Open source software. Pod kategorii Open source spadají základní knihovny EXAMu nikoliv samotný nástroj. [2]

To mimo jiné znamená, že je zdarma ke stažení ale zdrojové kódy samotného nástroje nejsou veřejně dostupné [3]. Od toho se také odvíjí, do jaké míry jsou v obsahu této práce popsány a zveřejněny detaily vývoje softwaru.

Software pro automatizaci testů EXAM je řešení a metodologie pro grafický vývoj testovacích případů (test case) založený na Unified Modeling Language (UML). Kromě hardware-in-the-loop (HIL) simulací a automatizace testovacích stolů (test bench) patří mezi oblasti využití například průmyslová automatizace, vývoj vestavných (embedded) systémů a software-in-the-loop (SIL) simulace. [4]

Mezi účely testovacího softwaru EXAM patří například snaha urychlit samotný proces testování. Z toho vyplývá lepší využití prostředků ve formě vytíženosti HILů. Zároveň toto řešení automatizace uleví zaměstnancům, zatímco udrží nízké náklady. [4]

Další výhodou využití modelování testovacích sekvencí v EXAMu je jejich nezávislost na cílové platformě. Jeden a ten samý test jde díky tomu spustit na různých platformách. Jen je třeba pro test vybrat příslušnou konfiguraci. Díky jednoduchosti grafického modelování testovacích sekvencí, není vyžadována profesionální znalost programování. [2]

1.1.1 Hardware in the loop

Sousloví Hardware in the loop (hardware ve smyčce) lze popsat akronymem HIL. HIL je způsob testování, ve kterém se reálné signály přivádějí do systému, jenž simuluje realitu. To způsobuje, že se řídící jednotka nachází ve stavu, který je z jejího pohledu shodný s konečným vestavěním v produktu. Iterace testování a návrhu pak probíhají jako by byl použit reálný produkční systém. Díky tomu je možné provést tisíce testů s různými scénáři ještě před tím, než se řídící jednotka zabuduje do produktu. To výrazně snižuje náklady spojené se skutečným fyzickým testováním systémů. [5]

Hardware in the loop test může nahradit například motor vozidla simulací složenou z hardwaru a softwaru, který interaguje se skutečnými vstupy a výstupy, jako kdyby se jednalo o skutečný motor. Díky možnosti provádět aktualizace v softwaru je možné rychle zařadit nové změny softwaru ECU nebo motoru a testovat chování řízení s různými okolnostmi, a to bez potřeby riskovat poškození drahého fyzického vývojového systému. [5]

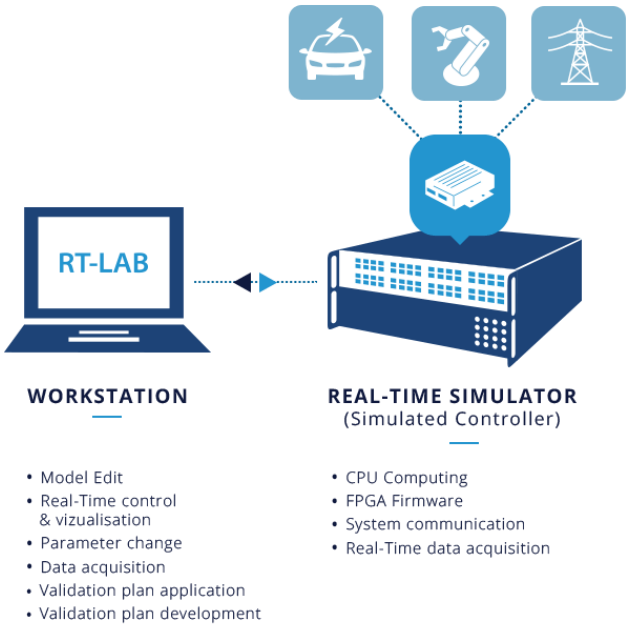
Obrázek 1: HIL NovaCarts Engine Ultimate [6]

1.1.2 Software in the loop

Sousloví Software in the loop (software ve smyčce) lze popsat akronymem SIL. SIL simulaci lze chápat jako integraci zkompilovaného produkčního kódu do simulace matematického modelu. Tato integrace poskytuje uživatelům praktickou virtuální simulaci pro vývoj a testování řízení rozsáhlých a komplexních systémů. [7]

Díky SIL mohou uživatelé využívat PC pro přímé iterativní testování a pozměňování zdrojového kódu. Tohoto je možné dosáhnout díky přímému spojení řídícího softwaru s digitálním modelem, který se používá jako náhražka za dražší fyzické systémy nebo prototypy. SIL umožňuje testování softwaru ještě před zahájením vývoje prototypů, což výrazně zrychluje celkový vývojový cyklus produktu. [7]

Mezi výhody využití SIL testování patří brzké odhalení nedostatků a chyb na systémové úrovni. To snižuje náklady na odstraňování problémů v pozdějších stádiích vývoje, kdy je obecně zvyšuje složitost a vnitřní provázanost problému. [7]



Obrázek 2: Schéma SIL a jeho částí [7]

1.1.3 Datová struktura v programu EXAM

Pro jasnější popis práce na softwaru EXAM je třeba upřesnění různých pojmů a užívaných datových struktur. Informace o nich jsou mimo jiné poskytnuty skrze dokument help v samotném programu.

V EXAMu jsou čtyři hlavní strukturní elementy, které se používají pro modelování testů. Prvním je TestCase. Ten slouží pro modelování testovacích případů. Tyto případy se dají dále pozměňovat a parametrizovat pomocí testovacích sekvencí (TestSequence) a testovacích aktivit (TestActivity). Testovací případy jsou spouštěny skrze testovací sadu (TestSuite). Druhým strukturním elementem je již dříve zmíněná testovací sekvence. Testovací sekvence jsou využívané pro rozdělení testovacích případů do různých částí. Tyto části pak mohou být jednoduše využitelné v jiných testovacích případech. Testovací sekvence popisuje tok informací mezi jednotlivými objekty v chronologickém pořadí a je v grafickém rozhraní orientována shora dolů. Další element je testovací aktivita, která také tvoří různé části testovacích případů, avšak má odlišnou strukturu oproti testovací sekvenci. Testovací aktivita je popsána pomocí takzvaného diagramu aktivit, jehož hlavní vlastností je možnost paralelního výkonu operací. Poslední strukturní prvek se nazývá administrativní případ (AdministrativeCase). Ten slouží pro přizpůsobení vykonávaného testu pro určitý HIL simulátor. Administrativní případy zprostředkovávají návrh inicializačních sekvencí, deinicializačních sekvencí a výjimkových sekvencí. Stejně jako testovací případy jsou i administrativní případy volány testovací sadou, která spojuje dohromady jednotlivé části testu. Zároveň se skrze ni samotný test spouští.

Základním stavebním blokem testovacích případů a testovacích sekvencí jsou operace. Ty jsou vytvářeny jako části implementační třídy (ImplementationClass) a mají svůj protějšek v takzvaném rozhraní (interface). Operace v implementačních třídách jsou vytvářeny jako psaný kód v jazyce Python (Python Body), nebo v podobě sekvenčního diagramu. Pro každou implementační třídu se pak automaticky generuje Python kód, který obsahuje třídu, v níž jsou definované metody, které představují uživatelem vytvořené operace.

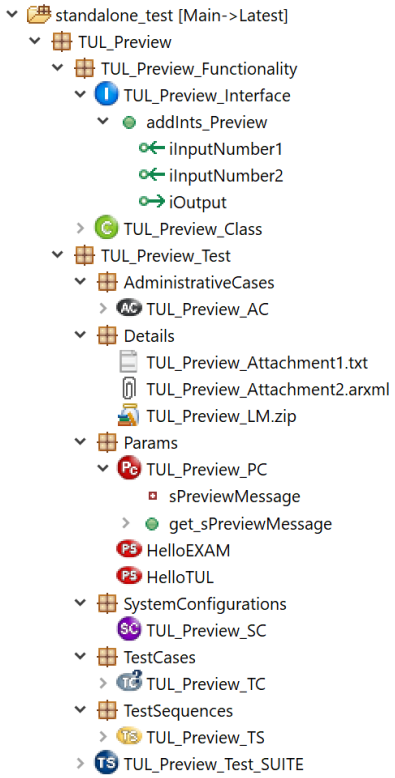
V testovacích případech a sekvencích se pak zpravidla volají operace z rozhraní nikoliv napřímo z implementační třídy, pokud je to možné. Jedno rozhraní může být implementováno pomocí několika tříd. Spojení určitých tříd s použitými rozhraními se provádí pomocí systémových konfigurací (SystemConfiguration).

Pro parametrizaci testovacích případů a sekvencí se používají parametrizační třídy. Ty obsahují parametry a k nim spjaté „get“ operace, které se volají pro zpřístupnění parametru v testovacím případu nebo sekvenci. Naplnění parametrů různými daty se provádí pomocí sady parametrů (ParameterSet), která se spojuje s daným testovacím případem nebo sekvencí, a tím se vybírá nastavení spuštěného testu.

Do kategorie entit využitelných při návrhu testovacích případů a sekvencí patří i takzvané přílohy. Ty se dělí na přílohy obecné a speciální knihovní přílohy. Obecné přílohy mohou být nejrůznější datové typy, se kterými je třeba v testu pracovat, například soubor ARXML (AUTOSAR Extensible Markup Language), jenž může obsahovat popis CAN (Controller Area Network) sítě v automobilu. Knihovní moduly (LibModule) jsou speciální knihovní přílohy, které skýtají Pythonovské moduly obsahující knihovnu nebo skupinu knihoven. Samotné Pythonovské moduly nelze ukládat přímo do EXAM modelu, a proto je třeba k nim přistupovat jako k příloze. Tato příloha je v podobě zip souboru s definovanou vnitřní strukturou.

Obecně vzato obsahují testovací případy takzvané kroky. Takový krok může reprezentovat například výše zmíněné operace nebo ostatní objekty popisující jiné sekvenční diagramy jako například testovací případ nebo sekvence.

V EXAM modelu existují dva typy modelových domén (ModelDomains): knihovna (Library) a testovací logika (TestLogic). Modelové domény rozdělují samotný model do funkčních množin a umožnují přiřazení určitých skupin uživatelů k jejich obsahu. V podstatě všechny EXAM prvky jsou uložené v modelové doméně. Rozdělení prvků uvnitř domén se provádí skrze využití takzvaných balíčků (package). Všechny objekty v EXAMu mají své unikátní ID (identifikátor). Na další straně je vyobrazena ukázka základní datové struktury pro testování v programu EXAM.



Obrázek 3: Ukázka datové struktury softwaru EXAM

1.1.4 Sekvenční diagramy

Pro sestavení jednotlivých testovacích případů a sekvencích pro HIL nebo SIL testování se v softwaru EXAM využívá tvorby sekvenčních diagramů. EXAM staví své sekvenční diagramy na základě univerzálního modelovacího jazyka UML (Universal Modeling Language). [2]

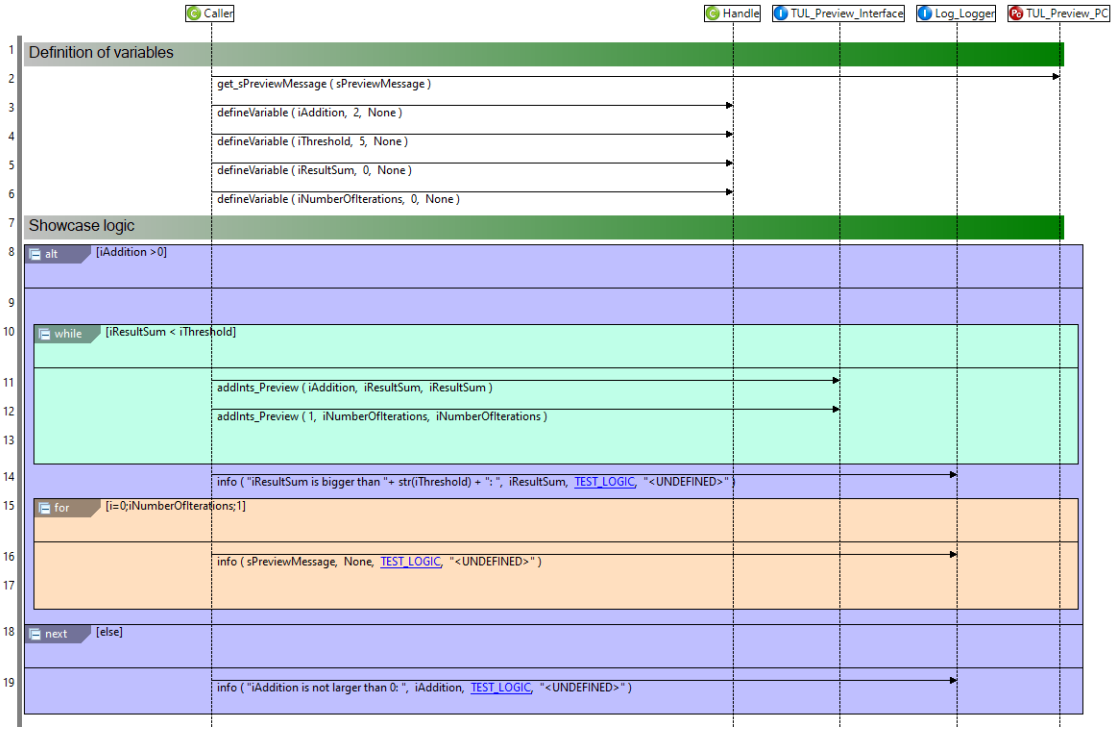
UML je standardním modelovacím jazykem pro vývoj softwaru a systémů. Jedním z důležitých důvodů pro modelování při návrhu systému je řízení složitosti. Modelovací jazyk může být sestaven z pseudokódu, opravdového kódu, obrázků nebo diagramů. Je to v podstatě cokoliv, co návrháři pomáhá lépe popsat určitý systém. Mezi hlavní výhody využití UML patří například následující vlastnosti. Každý prvek tohoto jazyka je jasně definován a snižuje tak možnosti nedorozumění při popisu určitých částí systému. Celý jazyk je sestaven z jednoduché a přímočaré notace, která popisuje všechny důležité aspekty systému. UML může být použit pro popis velkých komplexních systému, zároveň jej jde přiměřeně využít v malém měřítku. [8]

Další výhodou UML je značné posílení uživatelské přívětivosti programu EXAM. Díky využití UML je totiž možné modelovat sekvenční diagramy v testovacích případech a sekvencích relativně jednoduše a bez předchozí zevrubné znalosti jiného programovacího jazyka. Zároveň je pro testery, kteří se sekvenčními diagramy pracují,

jednodušší orientace a případné nacházení chyb nebo nedostatků v testovacích případech a sekvencích.

Sekvenční diagramy slouží k zobrazení souboru vzájemně se ovlivňujících objektů a posloupnosti zpráv, které jsou mezi nimi předávané. Sekvenční diagramy mohou také obsahovat doplňující informace o řídící struktuře během interakce těchto objektů. Mezi ně patří například if-else (pokud je a pravdivé pošli zprávu x, jinak pošli zprávu y), if-then (pokud je a pravdivé pošli zprávu x) a iterace (pošli zprávu n-krát). [9]

V softwaru EXAM jsou výše popsané zprávy reprezentovány jednotlivými operacemi rozhraní, které jsou implementovány pomocí metod psaných v jazyce Python. Objekty upravující chod operací jsou vyobrazeny pomocí takzvaných interakčních rámců. Sktruktura sekvenčního diagramu v EXAMu může být ovlivněna několika způsoby.



Obrázek 4: Ukázka sekvenčního diagramu v softwaru EXAM

Jedním z takových způsobů je například alt rámec, který se chová podobně jako příkaz if v programovacích jazycích. A protože jsou sekvenční diagramy vytvořené v EXAMu interpretované pomocí Python kódu, musí být podmínka, kterou alt obsahuje, platný Pythonovský příkaz. Mezi další interakční rámce patří alt_next, který je podobný příkazu elseif. Názvy ostatních rámců nejsou takto vzdáleny od svých protějšků v programovacích jazycích. Jsou to break, return a smyčky pro opakování určité části testovací sekvence while a for. Pro práci s výjimkami jsou k dispozici rámce catch, try a finally. Obrázek 4 je pouze okno sekvenčního diagramu, zobrazení celé aplikace EXAM je k dispozici v jako Příloha A.

1.2 Eclipse

Eclipse je univerzální softwarový vývojový nástroj s podporou pro více jazyků, který je využíván firmou MicroNova AG pro vývoj aplikace EXAM. Je to otevřené a rozšiřitelné integrované vývojové prostředí (Integrated Development Environment) neboli IDE. Toto prostředí je podporované vedoucími společnostmi v technologickém odvětví a dočkává se značného přijetí jak v komerčních, tak akademických kruzích v oblasti vývoje aplikací. [10]

Eclipse využívá takzvané bohaté klientské platformy (Rich Client Platform) neboli RCP. Bohatí klienti komunikující se serverovou stranou aplikace podporují pro koncového uživatele vysokou kvalitu uživatelské zkušenosti (User Experience – UX) se softwarem. Toho se dosahuje díky využití bohatých nativních uživatelských prostředí (User Interfaces), tedy UIs, a zároveň vysokorychlostních lokálních zpracování procesů. Bohatá uživatelská prostředí podporují nativní desktopové metafory. Mezi ně patří například funkce drag and drop, systémová schránka, navigace a přizpůsobení. Pokud je RCP zavedeno úspěšně, stává se bohatý klient pro uživatele skoro neviditelným. Tím zaměřuje pozornost uživatele na práci samotnou, a ne na mezivrstvu systému. Termín bohatého klienta byl vytvořen pro rozlišení mezi těmito klienty a klienty terminálovými nebo jednoduchými, které také nahradili. [10]

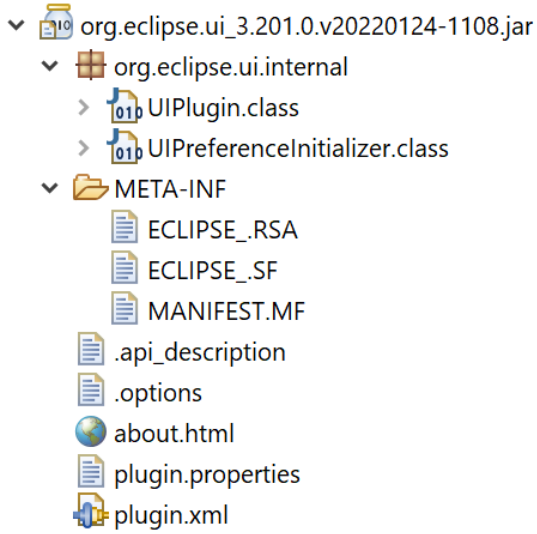
Pod integrovaným vývojovým prostředím pracuje nástrojová platforma podporující vícero programovacích jazyků a systémů, mezi které patří Java, C, Python, webové technologie, manipulace s daty nebo reportování. Pod touto nástrojovou platformou je samotné Eclipse RCP. To je obecná platforma pro spouštění aplikací. Jednou z těchto aplikací je Eclipse IDE. [10] EXAM je v tomto ohledu podobná aplikace jako Eclipse IDE, protože běží na stejné platformě, tedy na Eclipse RCP.

1.2.1 Plugin

Eclipse obsahuje robustní model komponent. Systémy, které jsou postaveny na Eclipse jsou sestaveny pomocí komponent, které se nazývají pluginy (plug-iny) neboli zásuvné moduly. Pluginy jsou verzovány a mohou být sdíleny mezi více aplikacemi. Různé verze jednoho pluginu mohou být nainstalované zároveň a aplikace, které tento plugin využívají mohou být nakonfigurované tak, aby využívali právě jednu jeho verzi. Tento přístup umožňuje aplikacím, aby se průběžně vyvíjely skrze přidávání a nahrazování komponent. [10]

Vše v Eclipse je plugin. RCP aplikace je souborem pluginů a rámcem, na kterém operují. RCP vývojář skládá dohromady soubor pluginů ze základny Eclipse a odjinud, přičemž přidává pluginy, které sám vytvořil. Tyto nové pluginy obsahují aplikaci a definici produktu zároveň s jejich doménovou logikou. Kromě pochopení, jak Eclipse spravuje pluginy, je zároveň důležité vědět, jaké pluginy je možné využít pro konkrétní případ a oproti tomu jaké pluginy je třeba sestavit vlastnoručně. Malé soubory pluginů jsou jednoduše spravovatelné, ale s rostoucím počtem pluginů v aplikaci je třeba využít skupinové abstrakce. Ta pomáhá skrýt část detailů systému. [10]

Plugin je soubor složek a manifestu, jež popisují plugin samotný a jeho vztah k ostatním pluginům. Na následujícím obrázku lze vidět příklad struktury pluginu org.eclipse.ui v prostředí Eclipse. První věc k povšimnutí je fakt, že samotný plugin je Java Archive (JAR). Jakožto JAR obsahuje plugin MANIFEST.MF. Tento manifest popisuje vlastnosti pluginu a jeho vztah s okolním světem. Pluginy mohou obsahovat jak kód, tak soubory určené pouze ke čtení. Tyto soubory mohou být například obrázky, webové stránky, dokumentace a jiné. [10]



Obrázek 5: Ukázka pluginu v Eclipse

Dalším zajímavým bodem je xml soubor uvnitř pluginu. Historicky to bylo místo pro informace spjaté s během programu, které jsou dnes ukládány do již dříve zmíněného MANIFEST.MF. Soubor plugin.xml nyní přetrvává jako místo pro rozšíření (extension) a deklarace bodů rozšíření (extension point), se kterými plugin operuje. [10]

1.3 Využité programovací jazyky

1.3.1 Java

Jako první je zde popsán programovací jazyk Java. Ten je využit k vývoji již dříve zmíněných pluginů v prostředí Eclipse pro aplikaci EXAM.

Programovací jazyk Java vznikl jako výzkumný projekt pro vývoj pokročilého softwaru pro nejrůznější síťová zařízení a vestavné systémy. [11]

Java nikdy nebyla pouhým programovacím jazykem. Je to celá platforma s rozsáhlou knihovnou obsahující spoustu znovu použitelného kódu, obsahuje také výkonné prostředí (execution environment), které poskytuje služby jako bezpečnost, přenositelnost mezi operačními systémy a garbage collection (automatický sběr odpadu). [12]

Java je od základu vybudována jako systém, který je relativně jednoduchý, objektově orientovaný, vícevláknový a interpretovaný. Díky tomu je Java pro většinu vývojářů snadno uchopitelná, využitelná v aplikacích klient-server, výkonná pro náročné aplikace, které vykonávají více úkonů naráz a je zároveň přenositelná a dynamická. [11]

V počátcích byla Java interpretovaný jazyk. Dnes využívá Java virtuální stroj, ve kterém se spouštěné programy zpracovávají, takzvaný just-in-time kompilátor. Díky tomu běží kritická místa kódu stejně rychle jako v jazyce C++, v některých případech může být Java dokonce rychlejší. [12]

1.3.2 Groovy

Groovy je volitelně typovaný, dynamický programovací jazyk pro platformu Java. Vývoj Groovy byl inspirován vícero jazyky, mezi které patří Python, Ruby nebo SmallTalk. Díky tomu Groovy zpřístupnil některé vlastnosti těchto jazyků vývojářům Javy skrze syntax podobnou Javě. Na rozdíl od ostatních alternativních jazyků se Groovy nesnaží nahradit Javu, ale být její společník a rozšířit její dosah. [13]

Groovy je často označován jako skriptovací jazyk, ale i když pro skriptování funguje dobře, nejde jej zredukovat jen na tuto funkci. Groovy může být předkompilován do Java bajtkódu, integrován do Java aplikací nebo power web aplikací. Zároveň může být základem pro celou aplikaci sám o sobě. Groovy je očividně příliš pružný na to, abychom ho dokázali jednoduše zaškatulkovat. [13]

Groovy je pevně spjatý s Java platformou. To platí jak pro implementaci (spousta částí Groovy je psána v Javě a zbytek je psán v Groovy samotném), tak pro interakci.

Když vývojář programuje v Groovy, píše ve speciálním druhu Javy a má pak k dispozici výhody, jako například rozsáhlý soubor knihoven, které s sebou přináší platforma Java. [13]

1.3.3 Python

Python je moderní programovací jazyk, pomocí kterého lze snadno vyvíjet jednoduché programy. Zároveň díky zpřístupnění výkonných prostředků dovoluje návrh poměrně rozsáhlých programů, jelikož má vývojář možnost využití obrovského množství knihoven. [14]

S rostoucí popularitou se Python stává klíčovým jazykem zejména v oblastech, které se soustředí na výuku a výzkum. Je to nejčastěji vyučovaný první jazyk na univerzitách, je nejpoužívanějším jazykem ve statistice, programování umělé inteligence, skriptování a psaní systémových testů. [14]

Python je založen na představě objektově orientovaného programování a patří mezi dynamicky typované jazyky. To mimo jiné znamená, že vše v něm je objekt a datový typ proměnných se vyhodnocuje až za chodu programu. [14]

1.3.4 SQL

SQL je jazyk, který zprostředkovává práci s relačními databázemi, tedy databázemi založenými na relačním modelu. Už při vzniku relačního modelu bylo základní myšlenkou, aby byla data reprezentována jako sada tabulek. To znamená, že místo adresování souvisejících entit pomocí ukazatelů jsou použita nadbytečná data spojující záznamy z různých tabulek. [15]

SQL jde ruku v ruce s relačním modelem také proto, že výsledkem SQL dotazu je tabulka (také nazývaná sadou výsledku – result set). Díky tomu je možné v relační databázi vytvořit novou permanentní tabulku pouze uložením výsledku předchozího dotazu. Podobným způsobem může dotaz využívat jak permanentní tabulky, tak sady výsledků z ostatních dotazů jako vstupy. [15]

V IT oboru jsou databáze velmi často ve středu dění, a proto je třeba je řádně navrhnout. Tématem návrhu databází se zabývá obor návrhové teorie (design theory). Ten se nezabývá pouze logickým návrhem základních datových struktur, ale i dalšími oblastmi. [16]

1.4 Data mining

Snahou data miningu (těžby dat) je získat případně užitečné znalosti z dat uložených v databázích, datových skladech nebo jiných informačních uložištích. [17]

Data mining se liší od tradiční statistiky v tom, že statistické odvozování je založené na předpokladech ve smyslu hypotézy tvořené a ověřené na základě dat, data mining je oproti tomu založen na objevech ve smyslu získání vzorů a hypotéz získaných přímo z dat. Jinými slovy, data mining je založen na datech, zatímco statistika na lidském vstupu. [17]

Jednou z nejdůležitějších oblastí data miningu je association rule mining (těžba sdružujících/asociačních pravidel). Association rule mining byl převážně vyvinut jako nástroj pro rozpoznání vztahů silně sdružující itemsety (sady položek), které mají velkou frekvenci a silnou souvztažnost. Asociační pravidla nám dovolují odhalit položky, které se v určité aplikaci často společně vyskytují. [17]

2 Vývoj softwaru

V této kapitole je popsáno a znázorněno navržené softwarové řešení problému. Je zde po částech popsán postup, pomocí kterého bylo dosaženo konečného výsledku v softwaru EXAM.

Tento postup se dělí na dvě hlavní části. První část se zabývá získáním a zpracováním relevantních dat z EXAM modelu a druhá část obsahuje přístup k výsledkům zpracování dat a jejich využití. Toto obecné rozdělení postupu je v následujících podkapitolách dále rozděleno a detailně popsáno.

První část řešení byla naprogramována pomocí dvou skriptů. První skript, který se stará o získání relevantních dat z EXAM modelu, je vytvořen přímo ve zpracovávaném EXAM modelu pomocí programovacího jazyka Groovy. Data získaná pomocí Groovy skriptu jsou jím také uložena do souboru ve formátu CSV. Druhý skript byl vytvořen již mimo EXAM model pomocí programovacího jazyka Python. Tento skript čte pro něj připravená data z CSV souboru a zpracovává je pomocí data mining operací. Jeho výstupem je rovněž soubor ve formátu CSV, ten obsahuje vytvořená pravidla a jejich vlastnosti, která jsou určena k dalšímu zpracování.

Zpracování pravidel vytvořených Python skriptem proběhlo na platformě Eclipse a v podobě pluginu. Eclipse plugin byl vyvinut pomocí programovacího jazyka Java. Tento plugin je schopen vytvořit lokální relační databázi a tabulky v ní a nahrát do ní pravidla vytvořená Python skriptem. Je zároveň schopen tuto databázi využít pro vyhledávání relevantních návrhů při tvorbě konkrétního diagramu, jehož obsah je v hledání zohledněn. Tyto návrhy plugin nakonec prezentuje skrze uživatelsky přívětivé prostředí uvnitř softwaru EXAM.

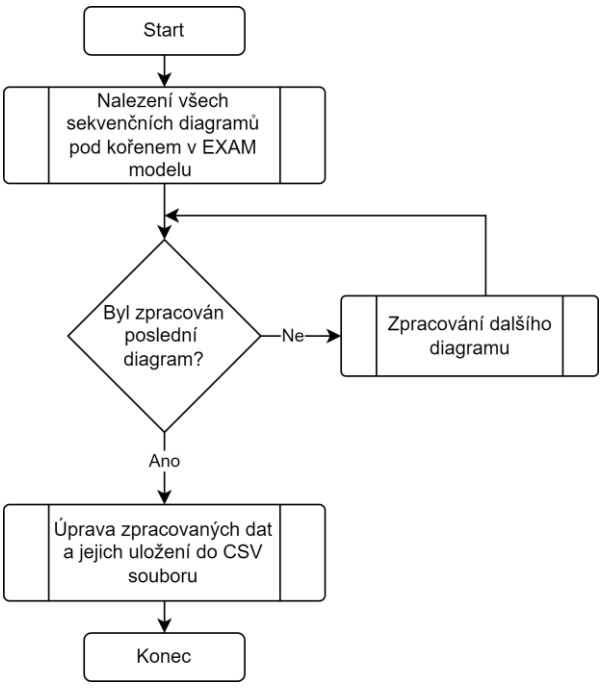
2.1 Získání základních dat

Pro sestavení modelu, na jehož základě bude realizována nápověda konkrétních elementů, je třeba nejdříve získat relevantní data, ze kterých bude model vycházet.

Pro sběr těchto dat byl vytvořen Groovy skript. Tento přístup byl zvolen z důvodu nativně podporovaného vývoje Groovy skriptů aplikací EXAM, pomocí kterých lze poměrně přímočaře provádět různé operace nad objekty v určitém EXAM modelu.

Každý Groovy skript v EXAMu jde spustit dvěma způsoby. První možnost je spustit skript sám o sobě z místa, kde se nachází. Druhý způsob spuštění je nejdříve zvolit určitý objekt, na který se má tento skript aplikovat. To, nad jakými objekty v EXAMu se dá skript spustit, se definuje ve vlastnostech konkrétního skriptu. V případě zde popisovaného skriptu byly nastaveny objekty knihovna (Library), balíček (Package) a testovací logika (TestLogic), které jsou popsané v kapitole 1.1.3.

Tato možnost dvojího spouštění se v samotném skriptu projevuje tak, že se v jeho základní třídě inherentně vytváří dvě metody, které se volají při spuštění skriptu, přičemž metoda spouštěná skrze výběr konkrétního EXAM objektu obsahuje parametr navíc. Tento parametr vyjadřuje ID (identifikátor) objektu, ze kterého je skript spouštěn. Ve zde popisovaném skriptu byly implementovány obě metody obdobně s tím rozdílem, že v jedné se jako kořen prohledávání bere objekt zadaný skrze parametr metody a v druhé je teprve vyhledáván počátek celého EXAM modelu, ve kterém se skript nachází. Další postup je stejný pro obě metody a jeho základní strukturu znázorňuje následující obrázek.



Obrázek 6: Základní popis Groovy skriptu

2.1.1 Datová struktura uložených dat

Před popisem postupu získání hledaných dat je nejdříve třeba vyjasnit podobu datové struktury, ve které budou data uložena. Podoba těchto dat je přímo odvislá od způsobu jejich zpracování (viz kapitolu 2.2.1), jenž očekává data v určité podobě. Nalezená data jsou ukládána do seznamu seznamů, který si lze představit jako tabulku, ve které první sloupec popisuje identifikátory všech nalezených sekvenčních diagramů a první řádek odpovídá identifikátorům všech unikátních kroků ve všech nalezených sekvenčních diagramech. Obsahem tabulky mezi prvním řádkem a sloupcem pak je binární vyjádření vztahu konkrétních diagramů a kroků. To znamená, že tabulka zahrnuje informace o tom, jestli daný sekvenční diagram obsahuje kterýkoliv z celkově nalezených kroků, přičemž hodnota „1“ vyjadřuje pozitivní vztah a „0“ vztah negativní. Takto uložená data jsou sice relativně objemná, ale jsou použitelná v dalším postupu. Tabulka 1 reprezentuje tuto strukturu na příkladu části již uložených dat.

	I4009	I1116287	I4505	I4559	I4470	I4509	I4071	I4526
I7059054	1	0	1	0	0	1	1	1
I7059098	0	1	1	1	1	0	0	0
I7059079	0	1	0	0	0	0	0	1
I7059106	0	1	0	0	0	0	0	0
I7060116	0	0	1	0	0	0	1	1
I7059072	0	1	0	0	0	0	0	0
I7059110	1	0	0	0	1	0	0	0
I7059058	0	1	0	0	0	1	0	0
I7059117	0	1	0	1	0	0	0	0
I7058055	0	1	0	0	1	0	1	0

Tabulka 1: Příklad části uložených dat získaných z EXAM modelu

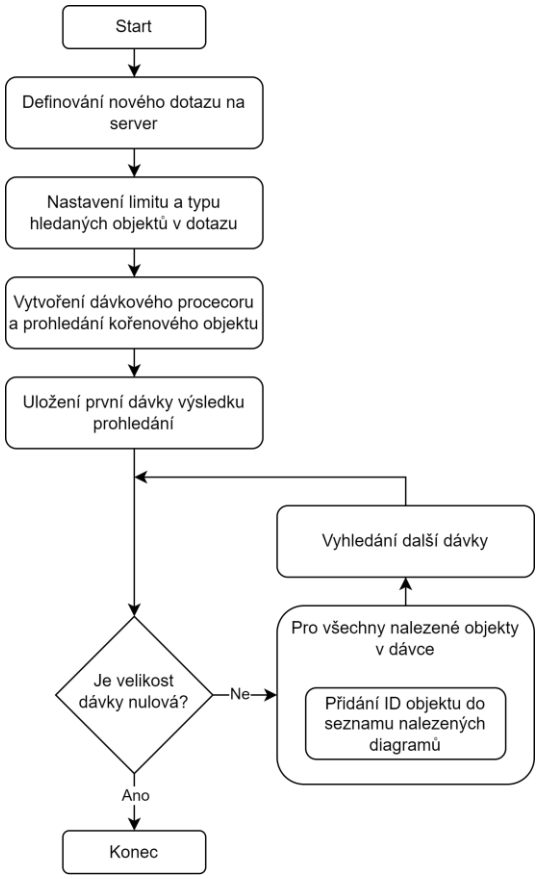
2.1.2 Přístup k EXAM modelu

Samotné získání dat proběhlo za využití již zmíněných startovacích metod, které využívají další tři metody, z nichž dvě data získávají a třetí je ukládá. Při popisu metod bude využito odkazů na názorný příklad struktury již uložených dat, které zobrazuje Tabulka 1.

První z těchto metod obstarává získání identifikátorů všech sekvenčních diagramů pod kořenem hledání, jehož identifikátor metoda přijímá jako parametr. V této metodě bylo nejprve třeba vytvořit prázdný seznam odpovídající prvnímu řádku v tabulce, který se následně doplnil o první záměrně prázdný záznam.

Dále se uskutečňuje vytvoření nového dotazu na server. Tento dotaz je třeba blíže specifikovat, a tak následuje nastavení limitu a typu hledaných objektů. Nastavený limit musí obsáhnout všechny potenciálně nalezené sekvenční diagramy, a proto je nastaven na řádově desítky tisíc. Typ hledaných objektů byl nastaven pomocí vnitřních tříd EXAMu na SEQUENCE_DIAGRAM, tedy pro hledání sekvenčních diagramů. Pomocí dotazu a identifikátoru kořenu, odkud má prohledávání začít, byl vytvořen dávkový procesor, ten byl obratem využit pro nalezení první množiny diagramů (dávky) hledaných objektů.

Po prvním prohledání následuje smyčka, která se vykonává, dokud aktuálně nalezená dávka není prázdná. Tato smyčka obsahuje vnitřní smyčku, která prochází všechny objekty v aktuální dávce a pro každý takový objekt vytvoří nový seznam s jedním záznamem vyjadřujícím identifikátor nalezeného objektu. Nově vytvořený seznam je pak přidán do základní datové struktury popsané výše (vloží se nový řádek do tabulky). Po dokončení vnitřní smyčky se vykoná příkaz pro vyhledání další dávky, vnější smyčka pak pracuje s novou dávkou výsledků. Tato činnost lze připodobnit k naplnění prvního sloupce tabulky 1 identifikátory všech nalezených sekvenčních diagramů. Popsaná metoda je obecně znázorněna následujícím vývojovým diagramem.



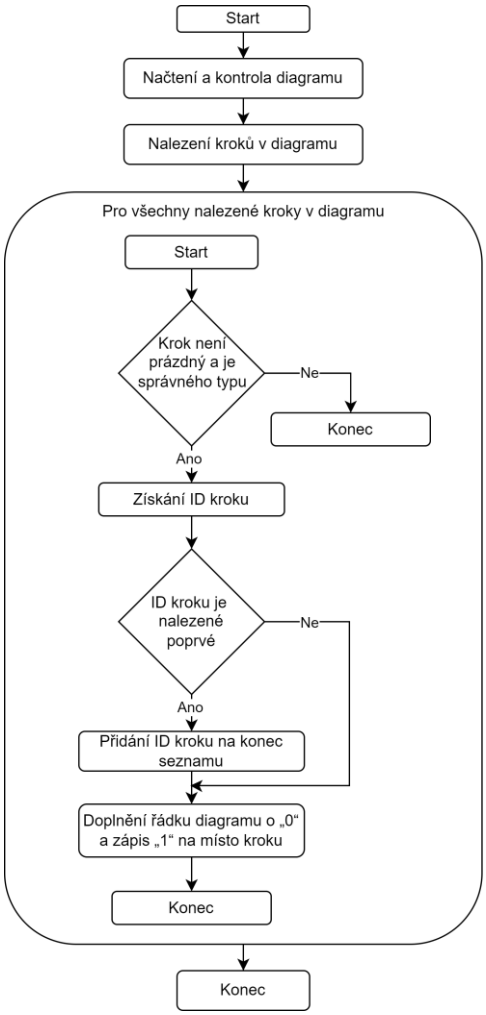
Obrázek 7: Metoda pro vyhledání sekvenčních diagramů z EXAM modelu

V prohledávání modelu přímo následuje další metoda, která je volána ve smyčce pro každý sekvenční diagram nalezený v předchozím kroku. Uvnitř metody je nedříve třeba načíst objekt diagramu, jehož ukazatel byl předán skrze parametr, zkontrolovat jeho typ a jestli není uzamčen jiným uživatelem. Poté se využije metoda, která prohledá daný sekvenční diagram a vrátí seznam všech kroků v něm. Tato metoda je součástí EXAM třídy sekvenčního diagramu.

Pokračuje smyčka, která provádí následující úkony pro všechny nalezené kroky. Nejprve je třeba ověřit, že krok má správný typ (tedy je to například volání operace, a ne například alt rámec, viz kapitolu 1.1.4) a není prázdný. Pokud jsou tyto podmínky splněny, proběhne získání identifikátoru obsahu kroku, přičemž je třeba dbát na rozdíl mezi kroky operací a kroky testovacích sekvencí. Tento identifikátor se přidá nakonec seznamu, který odpovídá prvnímu řádku tabulky 1, ale to pouze v případě, že ho seznam ještě neobsahuje.

Nakonec je třeba uložit do základní datové struktury informaci o tom, že je daný krok obsažen v právě procházeném diagramu. To probíhá způsobem, kdy se rozhoduje, jestli index sloupce, ve kterém se nachází ID kroku, je větší než (zaplněná) velikost řádku, který odpovídá prohledávanému diagramu. Pokud je podmínka splněna, je třeba tento řádek doplnit o hodnoty „0“ až do chvíle, kdy se narazí na sloupec s ID kroku. Na místo sloupce se vloží „1“, která vyjadřuje, že daný diagram obsahuje tento krok. Pokud předešlá podmínka není splněna, jinými slovy, že na daném místě, vyjadřujícím vztah diagramu a kroku, už je zapsaná nějaká hodnota, tak se na toto místo vloží hodnota „1“.

Na následující straně je pomocí vývojového diagramu vyobrazena základní struktura právě popsané metody.



Obrázek 8: Metoda pro zpracování sekvenčního diagramu

2.1.3 Úprava a uložení získaných dat

Před uložením získaných dat je třeba data lehce doupravit. Cílem těchto dat je vyjádřit obsah jednotlivých sekvenčních diagramů ve vztahu ke všem nalezeným krokům. Přesto toho nebylo do této chvíle dosaženo, protože takto úplný popis splňují pouze záznamy o diagramech, které obsahují poslední unikátní nalezený krok. Je tedy třeba doplnit ostatní seznamy (řádky) o samé hodnoty „0“.

Nyní jsou data připravena k dalšímu použití, a tudíž i k uložení. Pro uložení získaných dat byl vytvořen soubor typu CSV (comma-separated values). Ukládání do souboru probíhá pomocí dvou vnořených cyklů, ve kterých se přistupuje k jednotlivým elementům již dříve zmíněného seznamu seznamů. Přičemž jednotlivé záznamy jsou odděleny takzvaným separátorem a řádky jsou odděleny speciálními znaky pro odřádkování (\r\n). Takto uložená data získaná z EXAM modelu mají řádově tisíce řádků i sloupců. Tabulka 1 je příkladem části takto uložených dat.

2.2 Využití dat k tvorbě pravidel

Data získaná a uložená pomocí Groovy skriptu je třeba dále zpracovat pomocí Python skriptu, který vytváří pravidla využitá v dalších krocích řešení k nápovědě relevantních elementů při tvorbě sekvenčních diagramů. Skript byl vyvinut v integrovaném prostředí Pycharm. Zpracování dat probíhá za využití dvou hlavních Python knihoven: pandas a MLxtend. Pandas je rychlý, účinný a flexibilní open source nástroj pro analýzu a manipulaci dat [18] [19]. MLxtend je open source knihovna implementující vícero zásadních algoritmů a služeb pro machine learning (strojové učení) a data mining (těžbu dat) [20].

Skript nejdříve načte data uložená Groovy skriptem v předchozím kroku. Pro načtení těchto dat je využita pandas funkce, jejíž výstup je takzvaný DataFrame, což je datová struktura dále používaná pro práci s načtenými daty. Tato datová struktura je dvourozměrná a má jmenovitě označené řádky a sloupce, proto je třeba při načítání dat předat funkci indexy pro označení os.

2.2.1 Nalezení častých itemsetů

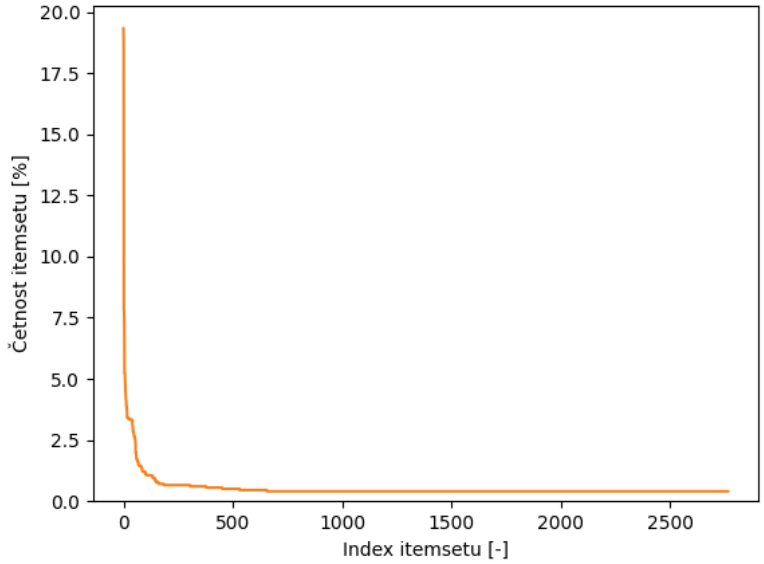
Po načtení dat následuje část kódu, pomocí které se nalézá alespoň určitý počet častých itemsetů (sad položek). Přičemž itemset v tomto kontextu vyjadřuje sadu kroků, které se společně vyskytují ve všech sekvenčních diagramech EXAM modelu s určitou četností. K nalezení těchto itemsetů byl využit algoritmus apriori, který je stejnojmenně implementován knihovnou MLxtend. Data předávaná této funkci musí mít určitou podobu, které je docíleno již při jejich vytvoření (viz kapitolu 2.1.1).

Nejzajímavějšími parametry využité implementace apriori jsou minimální četnost hledaných itemsetů, která (v tomto případě) výrazně ovlivňuje počet nalezených itemsetů, a maximální velikost hledaných itemsetů. Četnost itemsetů se v tomto kontextu označuje také jako podpora (support) a vyjadřuje, v jak velké části všech sekvenčních diagramů se itemset vyskytuje.

Aby bylo dosaženo nalezení alespoň určitého počtu itemsetů, je třeba opakovat hledání s postupně zmenšující se hranicí minimální četnosti pro hledané itemsety. Hledání se opakuje, dokud právě nalezená množina neobsahuje alespoň daný počet itemsetů a dokud je minimální četnost nastavena na pozitivní hodnotu. Z toho vyplývá, že výsledkem algoritmu sice může být množina menší než požadovaná, avšak tato množina bude ta největší možná při zadaném omezení velikosti itemsetů. Velikost

nalezené množiny itemsetů a itemsetů samotných je třeba takto kontrolovat kvůli zvyšující se výpočetní náročnosti při nabízení relevantních elementů v sekvenčních diagramech, protože nabídka je v důsledku založena na těchto itemsetech.

Pro zobrazení výsledků se získaná sada častých itemsetů seřadí podle četnosti a pomocí open source Python knihoven numpy a matplotlib se poté vykreslí jako graf. Takový graf vyjadřuje četnost itemsetu v závislosti na jeho indexu v seřazené sadě. Díky tomu je viditelné, jak rychle klesá četnost nalezených itemsetů a jejich počet v různých úrovních četnosti.



Obrázek 9: Graf seřazené četnosti itemsetů z vývojového EXAM modelu

Z grafu je zřejmé, že v použitém EXAM modelu existuje pár itemsetů, které mají výrazně vyšší četnost než ostatní. Tyto nejvyužívanější itemsety obsahují například operace pro vytvoření proměnné, nebo zaznamenání zprávy do informačního kanálu v průběhu testu.

2.2.2 Nalezení pravidel pro napovídání

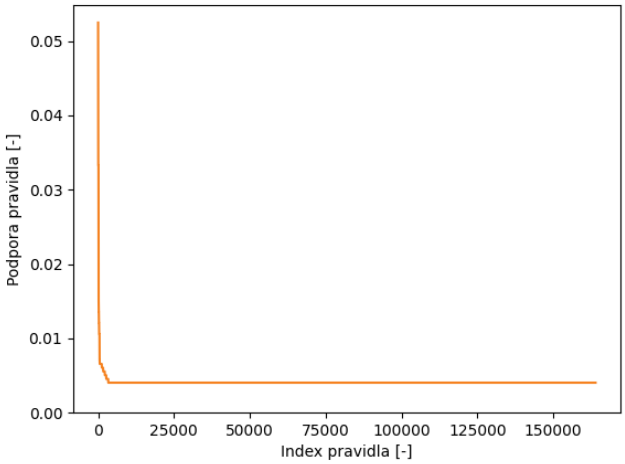
Po získání častých itemsetů z EXAM modelu se itemsety dále zpracovávají pomocí association rule mining algoritmu, který je implementován knihovou MLxtend pod názvem association_rules. Výstupem algoritmu jsou pravidla, která lze v použitém kontextu popsat jako spojení dvou množin kroků ze sekvenčních diagramů skrze implikační vazbu. Jinými slovy, pokud je v diagramu použita první množina kroků (předchůdci – antecedents), lze s určitými vlastnostmi předpokládat, že bude použita i množina druhá (následovníci – consequents).

Využité funkci je třeba předat vstupní data (časté itemsety), vlastnost k zohlednění a práh této vlastnosti. V následujících bodech jsou vyjmenovány vlastnosti, které jsou přiřazeny každému pravidlu. Jsou zde také vyobrazeny (obdobně jako v kapitole 2.2.1) grafy popisující průběh klesání těchto vlastností, přičemž vstupní data byla získána ze stejného EXAM modelu. Pokud budeme uvažovat předchůdce pravidla jako A, následovníky jako C a pravidlo jako A→C, pak lze vlastnosti pravidel popsat následujícím způsobem [21]:

- Podpora (support) již byla zmíněna v kapitole 2.2.1 jako četnost itemsetu v celém EXAM modelu. Při popisování pravidla se určuje podpora itemsetu předchůdců, itemsetu následovníků a samotného pravidla. Přičemž se podpora pravidla řídí následujícím vztahem a nabývá hodnot 0 až 1.

$$support(A \rightarrow C) = support(A \cup C)$$

Rovnice 1: Podpora pravidla



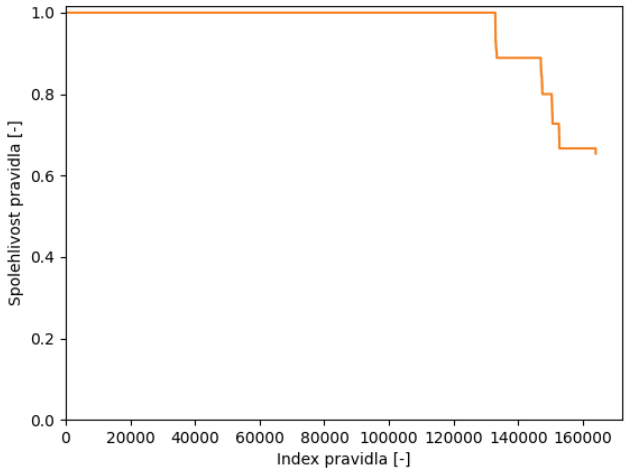
Obrázek 10: Podpora pravidel pro vývojový EXAM model

Z předchozího grafu je zřejmé, že četnost vytvořených pravidel se v přepočtu pohybuje pod 1 % a z vyšších hodnot klesá relativně rychle. Zdánlivě nízká podpora pravidel neznamená, že jsou pravidla nepoužitelná, pouze že EXAM model, ze kterého byly získány itemsety, je rozličný a rozsáhlý. Další vlastnosti blíže vypovídají o kvalitě a použitelnosti pravidel.

- Spolehlivost (confidence) vyjadřuje pravděpodobnost s jakou se v určitém sekvenčním diagramu nacházejí následovníci, za předpokladu, že se v něm již vyskytují předchůdci. Spolehlivost se řídí následujícím vztahem a nabývá hodnot od 0 do 1.

$$confidence(A \rightarrow C) = \frac{support(A \rightarrow C)}{support(A)}$$

Rovnice 2: Spolehlivost pravidla



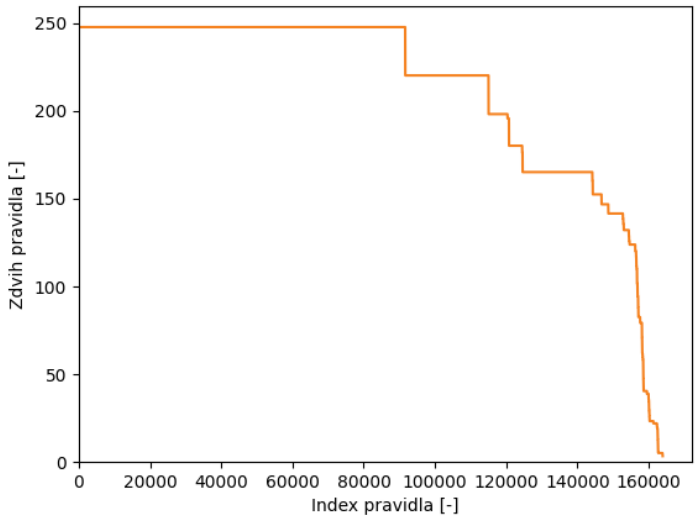
Obrázek 11: Spolehlivost pravidel pro vývojový EXAM model

Z grafu spolehlivosti je vidět, že zhruba tři čtvrtě pravidel jsou ohodnoceny ve spolehlivosti hodnotou 1 a mají tedy vysokou účinnost. Druhá část pravidel kaskádně klesá až k hodnotě 0,65, která byla na začátku zvolena jako krajní.

- Zdvih (lift) měří, jak často se vyskytují předchůdci a následovníci určitého pravidla společně oproti situaci, kdy by byli na sobě statisticky nezávislí. Tato vlastnost nabývá hodnot od 0 do ∞, přičemž nabývá hodnoty 1, když A a C jsou statisticky nezávislé. Zdvih je popsán následujícím vztahem.

$$lift(A \rightarrow C) = \frac{confidence(A \rightarrow C)}{support(C)}$$

Rovnice 3: Zdvih pravidla



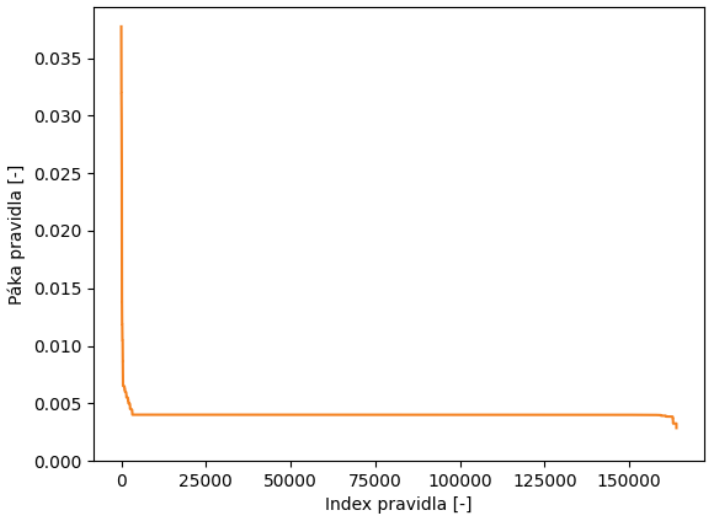
Obrázek 12: Zdvih pravidel pro vývojový EXAM model

Na grafu je jasně vidět, že většina hodnot zdvihu pravidel je vysoko nad hodnotou 1. Z toho lze usuzovat, že výskyt předchůdců v daném sekvenčním diagramu má pozitivní vliv na výskyt následovníků. Další zajímavou vlastností grafu je fakt, že klesá plynuleji než ostatní zde popsané. Této vlastnosti je možné využít při třídění pravidel pro konečný návrh elementů (viz kapitolu 2.3.2).

- Páka (leverage) udává rozdíl mezi četností, kdy se A a C vyskytují společně a četností při nezávislých A a C. Páka se řídí následujícím vztahem a nabývá hodnot od -1 do 1, přičemž hodnota 0 znamená nezávislost A a C.

$$leverage(A \rightarrow C) = support(A \rightarrow C) - support(A) \cdot support(C)$$

Rovnice 4: Páka pravidla



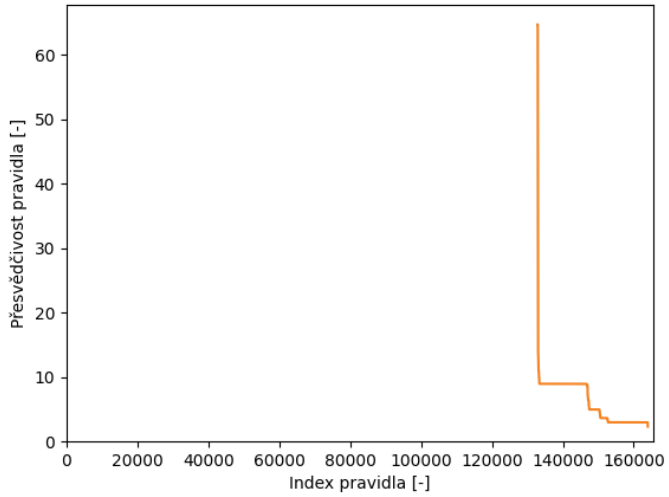
Obrázek 13: Páka pravidel pro vývojový EXAM model

Z grafu páky pravidel je zřejmá vlastnost všech pravidel, tedy že předchůdci a následovníci v nich jsou na sobě statisticky závislí. Tvar grafu se podobá tvaru grafu podpory pravidel.

- Přesvědčivost (conviction) nabývá hodnot od 0 do ∞. Její hodnota se zvyšuje, čím více je následovník pravidla závislý na předchůdci. Podobně jako zdvih nabývá hodnoty 1, když A a C jsou statisticky nezávislé. Přesvědčivost je popsána následujícím vztahem.

$$conviction(A \rightarrow C) = \frac{1 - support(C)}{1 - support(A \rightarrow C)}$$

Rovnice 5: Přesvědčivost pravidla



Obrázek 14: Přesvědčivost pravidel pro vývojový EXAM model

Vyobrazený graf přesvědčivosti zdánlivě začíná až okolo indexu pravidla 130 000. Je tomu tak, protože všechny předchozí hodnoty přesvědčivosti dosahují limitně nevykreslené hodnoty ∞. Kvůli tomu je průběh přesvědčivosti vlastně ještě strmější, než jak se může z obrázku zdát.

Protože cílem pravidel je poskytovat uživateli relevantní návrhy, byla pro nastavení limitu vlastnosti při jejich vytváření vybrána spolehlivost. Pomocí stanovení prahu spolehlivosti tak můžeme vybrat pravidla, pro která platí, že budou splňovat minimální pravděpodobnost, že navrhovaný element bude užitečný. Ve výše uvedených příkladech grafů byl nastaven práh spolehlivosti na 0,65.

Pokud se výsledná hodnota pravidla pro zdvih nebo přesvědčivost blíží k ∞, je ve výsledcích reprezentována jako string „inf“.

2.2.3 Úprava a uložení získaných pravidel

Pro uložení pravidel byl použit CSV soubor, obdobně jako v kapitole 2.1.3. Pokud by byla pravidla uložena přímo po jejich vytvoření, vypadal by uložený soubor následujícím způsobem:

	antecedents	consequents	antecedent support	consequent support
0	frozenset({'I15017'})	frozenset({'I75099'})	0.005549949545913219	0.01917255297679112
1	frozenset({'I15018'})	frozenset({'I75099'})	0.016145307769929364	0.01917255297679112
2	frozenset({'I2277_Pg'})	frozenset({'I14295'})	0.006559031281533804	0.19323915237134207
3	frozenset({'I15051'})	frozenset({'I75099'})	0.005549949545913219	0.01917255297679112
4	frozenset({'I15017', 'I15051'})	frozenset({'I15018', 'I75099'})	0.005549949545913219	0.015136226034308779

Tabulka 2: Příklad části neupravených pravidel

Kvůli dalšímu zpracování v databázi se data před uložením ještě upravují. Zaprvé se v názvech sloupců zamění mezery za podtržítka. Zadruhé jsou data obsažená ve sloupcích „antecedents“ a „consequents“ zbavena nadbytečných slov a znaků. Obsahují poté pouze identifikátory kroků a znaky pro jejich oddělení. Uložená pravidla pak vypadají takto:

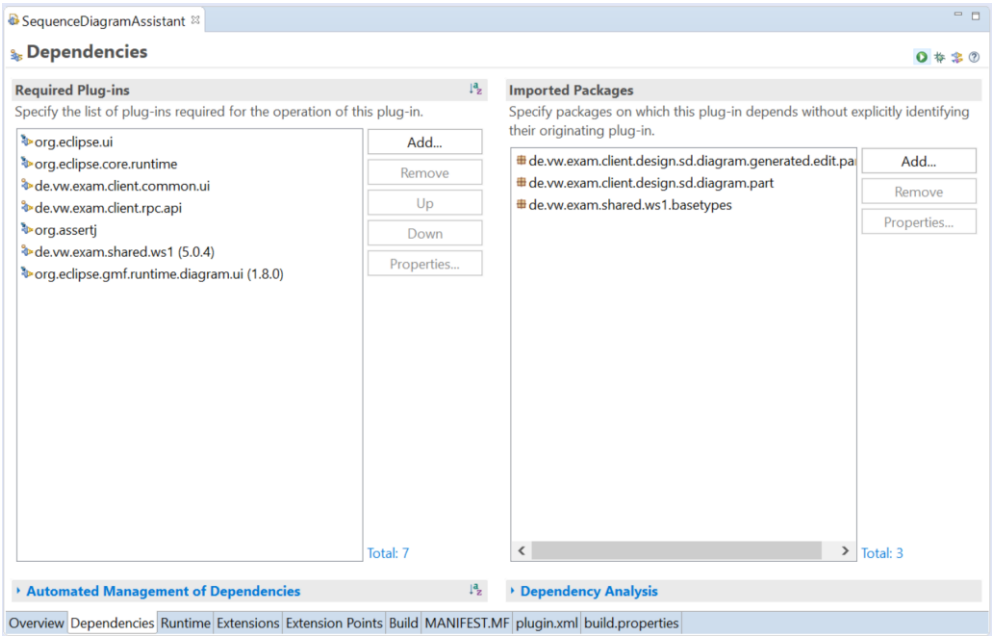
	antecedents	consequents	antecedent_support	consequent_support
0	I15017	I75099	0.005549949545913219	0.01917255297679112
1	I15018	I75099	0.016145307769929364	0.01917255297679112
2	I2277_Pg	I11625	0.006559031281533804	0.18718466195761857
3	I15051	I75099	0.005549949545913219	0.01917255297679112
4	I15051,I15017	I15018,I75099	0.005549949545913219	0.015136226034308779

Tabulka 3: Příklad části upravených pravidel

V tomto příkladu jsou vyobrazeny pouze čtyři pravidla, uložený soubor jich ale ve skutečnosti obsahuje až desetitisíce.

2.3 Tvorba Pluginu pro EXAM

Práce s uloženými pravidly probíhá skrze Eclipse plugin do aplikace EXAM, který je nejprve třeba vytvořit jako Plug-in Project. Plugin je poté třeba nastavit jako součást spouštěné aplikace a případně úroveň jeho spuštění. Dále je třeba v manifestu pluginu (viz kapitolu 1.2.1) specifikovat ostatní pluginy a balíčky, na kterých je vyvíjený plugin závislý. To se provádí v záložce „Dependencies“, jak je vidět na následujícím obrázku.



Obrázek 15: Nastavení závislostí pluginu

Zde je zřetelné, že vytvořenému pluginu byly přidány jak závislosti na obecné Eclipse pluginy, tak závislosti na proprietární pluginy a balíčky začínající „de.vw.exam“.

Obdobným způsobem bylo v záložce „Runtime“ přiřazena cesta k jar souboru, který obsahuje knihovnu pro práci s H2 databází (viz kapitolu 2.3.1). V záložce „Extensions“ byl přidán nový Extension Point (rozšiřující bod) typu „org.eclipse.ui.views“, který zajišťuje vytváření a práci s novými okny v Eclipse aplikaci. Tomuto bodu byla přidělena třída popisující vlastnosti a chování okna, jež plugin vytváří (viz kapitolu 2.3.3).

Vytvořený plugin implementuje tři třídy: Activator pro inicializaci a deinicializaci pluginu, H2DB pro práci s H2 databází a SDAssistantView pro práci s oknem zobrazujícím návrhy do sekvenčního diagramu.

2.3.1 Uložení pravidel do relační databáze

Pro práci s vytvořenými pravidly byla využita lokální relační databáze, konkrétně open source H2 SQL databáze [22]. K této databázi se přistupuje skrze JDBC API (Java database connectivity application programming interface). Díky tomuto API je možné v pluginu například vytvářet a plnit tabulky, nebo se databáze dotazovat skrze kód psaný v Javě.

Načtení dat do databáze probíhá hned při startu pluginu, což zajišťuje metoda start ve třídě Activator. V metodě start se volají dvě metody z třídy H2DB. První volaná metoda executeStatement zajišťuje provedení určitého dotazu na databázi, přičemž dotaz je popsán SQL syntaxí v argumentu metody jako string. V tomto případě je posílán dotaz pro odstranění tabulky pravidel, pokud existuje. Vzápětí je volána druhá metoda readRulesFromCSV, která vytváří tabulku pravidel a přidává do ní data vyčtená z CSV souboru.

Obě tyto metody musí nejdříve navázat spojení s lokání databází, čehož je dosaženo skrze další, tentokrát privátní, metodu třídy H2DB getConnection. Pro navázání spojení se nejdříve vytváří objekt connection pomocí rozhraní java.sql.Connection. Objekt connection je naplněn pomocí JDBC metody, která jako argumenty přijímá adresu databáze, uživatelské jméno a heslo pro připojení.

Po připojení k databázi se v metodě executeStatement vytváří tvrzení (statement), které se vzápětí provádí skrze jeho metodu execute, ta jako parametr přijímá SQL dotaz, který se má vykonat, jako string.

Metoda readRulesFromCSV pracuje obdobným způsobem s tím rozdílem, že základní struktura SQL dotazu je pevně daná a dynamicky se do ní doplňují informace o cestě k CSV souboru a názvu tabulky, která má být vytvořena. V SQL dotazu se také upřesňují odpovídající datové typy jednotlivých sloupců a separátor čteného CSV souboru.

2.3.2 Nacházení relevantních pravidel a návrhů

Pro nalezení relevantních pravidel určených k návrhu při tvorbě sekvenčních diagramů je ve třídě H2DB vytvořena metoda findSuggestionIDs.

Tato metoda přijímá jako argument seznam identifikátorů kroků, které již existují v daném sekvenčním diagramu a maximální velikost navrhované množiny kroků. Přičemž první z těchto kroků je považován za kořen pro hledání relevantních pravidel. Kořenem je v tomto kontextu myšlen krok, který byl do diagramu přidán jako poslední a má při prohledávání pravidel nejvyšší prioritu. Výstupem metody je rovněž seznam identifikátorů, v tomto případě ale kroků, které mají být uživateli navrhnuty.

Algoritmus pro nalezení relevantních návrhů probíhá následujícím způsobem. Nejprve se získá připojení k databázi obdobně jako v kapitole 2.3.1, pomocí kterého jsou vykonávány SQL dotazy. Protože jsou očekávány výsledky dotazů, které jsou v této metodě použity, je třeba vytvořit objekt (obdobně jako se vytvořil objekt connection) resultSet (sada výsledků), pomocí kterého se přistupuje k výsledkům dotazu.

Poté se vykoná dotaz, který hledá všechna pravidla, v jejichž předchůdcích se vyskytuje identifikátor kořenového kroku. Pokud byla taková pravidla v databázi nalezena, algoritmus pokračuje dál, a to tak, že uloží indexy pravidel, pro která byla podmínka splněna, jako seznam, který je využíván jako základ pro další prohledávání databáze.

Vzápětí se naplní nová hashmap, jejíž klíče jsou nastaveny na indexy nalezené v předchozím kroku, hodnoty odpovídající těmto klíčům jsou nastaveny ve výchozím stavu na 1 a představují ohodnocení daného pravidla.

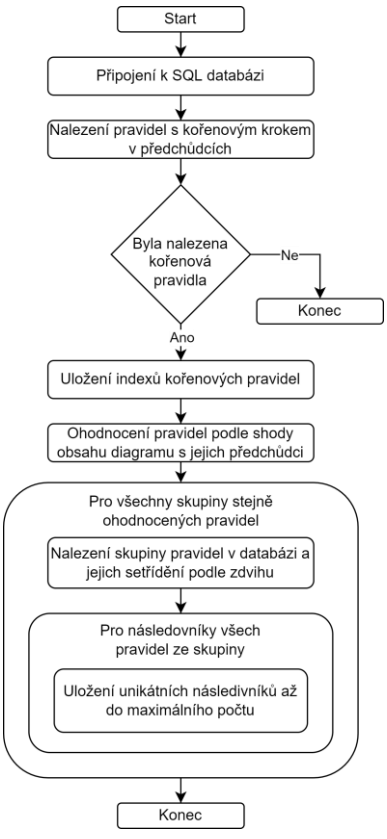
Dále následuje smyčka, která prochází identifikátory kroků přítomných v sekvenčním diagramu a pro každý takový krok vykoná dotaz k prohledání databáze, přičemž prohledávaná množina pravidel je omezena na výše popsaný základ celé množiny pravidel. V této omezené množině se hledají taková pravidla, jejichž předchůdci obsahují daný krok. Pro všechna nalezená pravidla se poté v hashmap zvýší jejich hodnota o 1. Touto smyčkou se docílí ohodnocení pravidel ze základní množiny, s tím že vyšší hodnota představuje vyšší počet kroků shodujících se s předchůdci pravidla a vyjadřuje vyšší prioritu pravidla při dalším hledání.

Po ohodnocení pravidel se vytvoří pole obsahující indexy a hodnoty pravidel. Pole je obratem setříděno podle hodnot pravidel. Takto setříděná pravidla jsou nejprve využita k naplnění nového seznamu, jenž vyjadřuje velikosti jednotlivých skupin pravidel podle

jejich hodnoty. Pravidla v jednotlivých skupinách mají při hledání relevantních pravidel stejnou prioritu, a tak se jejich velikost používá pro provedení posledního SQL dotazu vždy pro celou takovou skupinu.

Následuje tedy smyčka, ve které se dynamicky vytváří SQL dotaz, jež prohledává databázi pro pravidla specifikována částí setříděného pole dříve nalezených pravidel. Tato část odpovídá skupině pravidel se stejným ohodnocením. Dotaz zároveň setřídí nalezená pravidla podle vybrané vlastnosti pravidel (viz kapitolu 2.2.2). V tomto případě byl pro setřídění zvolen zdvih, protože, jak Obrázek 12 znázorňuje, průběh této veličiny je v porovnání s ostatními vlastnostmi relativně pozvolnější. Díky těmto vlastnostem je zde vhodnější zdvih než například spolehlivost nebo přesvědčivost, pomocí kterých nelze rozlišit větší část nalezených pravidel. Po nalezení a seřazení pravidel se prochází jejich následovníci s tím, že pokud již nejsou uloženi do seznamu navrhovaných kroků a seznam je menší než jeho maximální velikost určená v parametru metody, tak se přidají na jeho konec. Smyčka se poté opakuje pro další skupinu ohodnocených pravidel.

Výše popsaný algoritmus prioritizuje primárně pravidla, která obsahují ve svých předchůdcích poslední vložený krok do sekvenčního diagramu, sekundárně prioritizuje pravidla, která mají vyšší shodu kroků v jejich předchůdcích s kroky uvnitř diagramu, a nakonec seřadí tato pravidla podle jejich zdvihu.



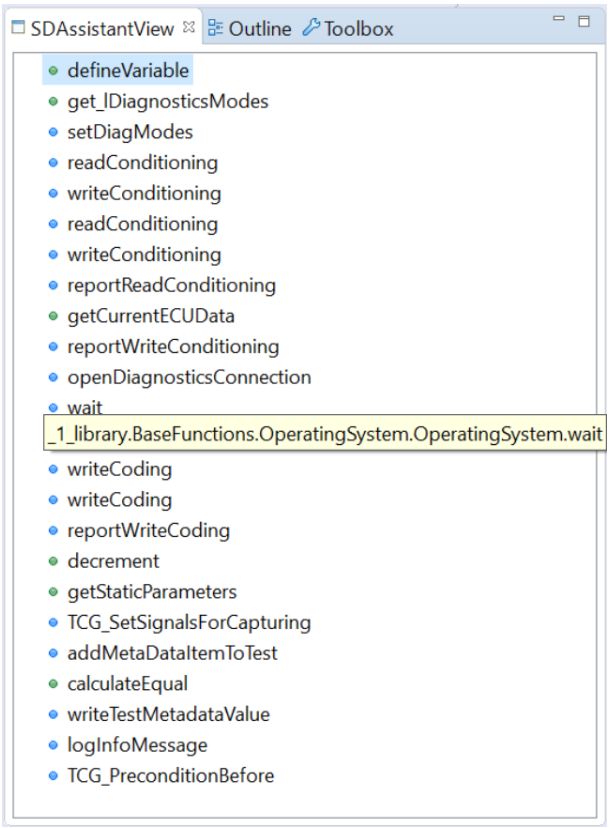
Obrázek 16: Nacházení relevantních pravidel a návrhů

2.3.3 Napovídání prvků uživateli

Pro interakci s uživatelem je zapotřebí UI (user interface – uživatelské prostředí), které bylo zprostředkováno pomocí nového okna uvnitř aplikace EXAM. Pro nové okno byla v pluginu vytvořena třída SDAssistantView, která je spjatá s jeho Extension Pointem. Tato třída rozšiřuje ViewPart, což je jedna ze základních tříd poskytnutých skrze platformu Eclipse. Nová třída překrývá metodu createPartControl, která se stará o funkčnost vytvořeného okna, zároveň obsahuje novou metodu suggestSDElements, která obstarává nalezení a vložení navrhovaných elementů do okna.

V metodě createPartControl se nejprve vytváří objekt treeViewer pomocí třídy TreeViewer, která je založená na SWT (standart widget toolkit) v Eclipse a popisuje strukturu dat zobrazovaných v okně. Po vytvoření objektu je třeba upřesnit informace o něm. Za prvé, jak má vyplňovat okno, kterému náleží. Dále se nastavuje podoba dat, které se pomocí něj zobrazují. Podoba zobrazovaných dat se nastavuje skrz implementaci Eclipse rozhraní ITreeContentProvider, ve kterém je stěžení metoda getElements, jež ze vstupních elementů (konkrétně z identifikátorů kroků) vytváří seznam objektů, které bude možné vložit do sekvenčního diagramu.

Následuje nastavení schopnosti objektu zobrazit takzvaný tool tip, což znamená zobrazení detailních informací ohledně objektu, na který právě uživatel míří myší. Tyto a další v okně zobrazované informace je třeba dále specifikovat, toho se dosáhne pomocí překrytí EXAM třídy pro poskytování označení. První překrytá metoda poskytuje string, který představuje popisek zobrazovaného objektu, v tomto případě jméno navrhovaného kroku. Druhá metoda taktéž vrací string představující text, který se zobrazí jako tooltip. Zde se jako tooltip vkládá celá cesta navrhovaného kroku. To může například pomoct uživateli rozlišit mezi stejnojmennými kroky, které jsou však implementovány na různých místech v EXAM modelu.



Obrázek 17: Okno s návrhy a zobrazeným tool tipem

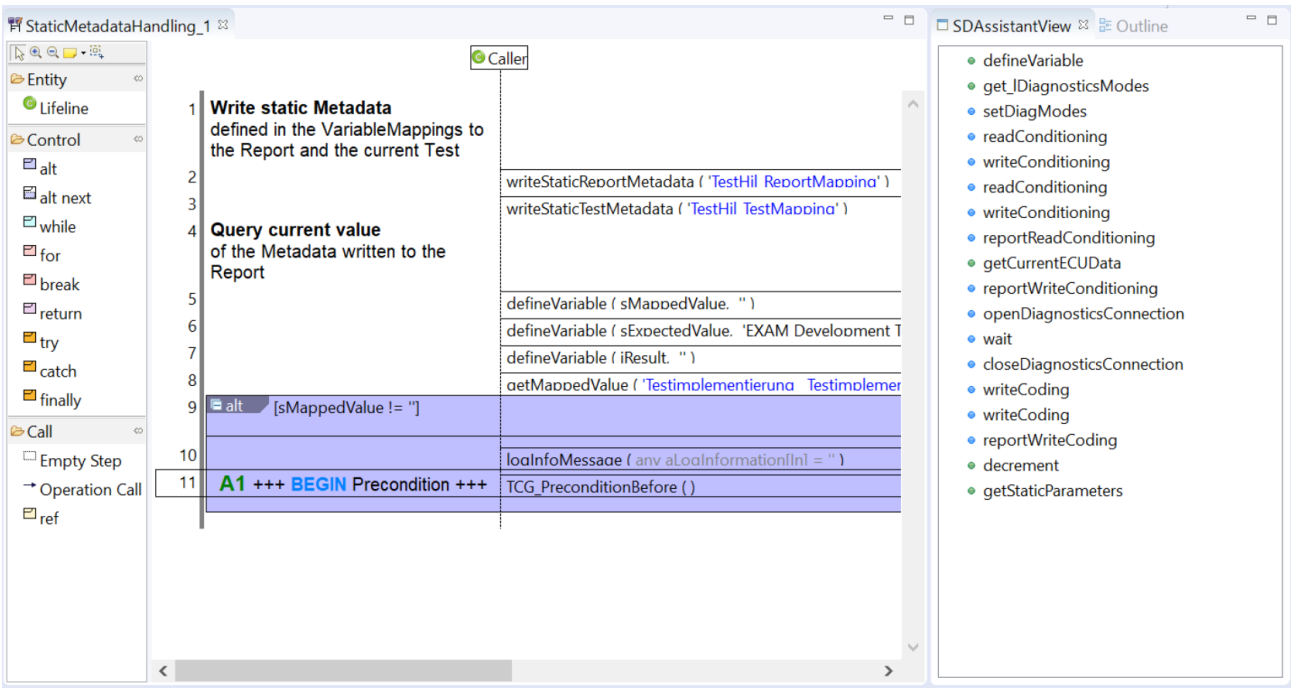
Dále se pomocí SWT prostředků nastavuje podpora pluginu pro takzvanou drag and drop funkcionalitu. Relativní jednoduchost zavedení této funkcionality je jednou z výhod využití Eclipse RCP (viz kapitolu 1.2).

Po nastavení vlastností zobrazovaných dat je třeba určit v jaké chvíli se bude přistupovat k databázi a co bude předáváno jako vstupní data pro vyhledávání v ní. Pro to se nejprve vytváří nový objekt, který sleduje změnu uživatelem vybraného objektu v rámci aplikace EXAM. Bylo třeba implementovat metodu vytvořeného objektu, která se spouští při takové změně, a to tak, že se nejprve kontroluje typ vybraného objektu, tedy že se jedná o krok v sekvenčním diagramu. Poté se získají všechny ostatní kroky v tomto diagramu a zavolá se metoda suggestSDElements se seznamem získaných kroků v argumentu.

Metoda suggestSDElements nejprve získá seznam identifikátorů kroků, které se v diagramu právě vyskytují, získá rozdíl toho seznamu se seznamem kroků, které byly v diagramu při předchozím volání metody. Pomocí rozdílu se rozhodne, jestli se od minulého volání metody diagram změnil, pokud ne, algoritmus nepokračuje. Vzápětí se rozdíl použije pro přemístění naposledy vloženého kroku na poslední místo seznamu aktuálních kroků v diagramu, jelikož je tento krok považován za kořen hledání relevantních pravidel (viz kapitolu 2.3.2).

Nyní je možné ze třídy H2DB volat metodu findSuggestionIDs (viz kapitolu 2.3.2) pro nalezení relevantních pravidel odpovídajících aktuálně zpracovávanému sekvenčnímu diagramu. Seznam identifikátorů vrácený touto metodou se uloží do nového seznamu, ke kterému se vzápětí připojí identifikátory kroků z minulého hledání. Protože tyto kroky nejsou stejně relevantní jako ty právě nalezené, tak se přidávají nakonec seznamu, tedy jsou uživateli zobrazeny vespod nabídky. Aby uživateli nebyl představován jeden krok vícekrát, je třeba seznam zbavit duplikátů, přičemž zůstávají kroky, které se v seznamu vyskytují dříve. Odeberou se tedy duplikáty s nižší prioritou.

Na konci metody se rozhodne o tom, zda není seznam nabízených identifikátorů prázdný. Pokud není, nastaví se seznam jako vstupní data pro zobrazení výsledků uživateli.



Obrázek 18: Okno pro napovídání společně s vyvíjeným diagramem

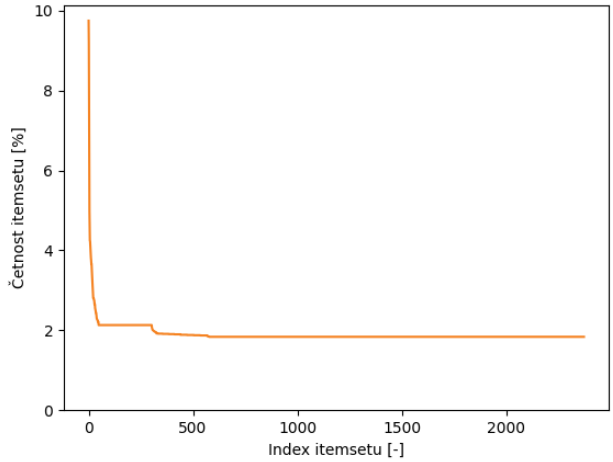
Na obrázku výše je vidět ukázka možného použití pluginu při tvorbě konkrétního sekvenčního diagramu. Okno napravo zobrazuje rozšířené návrhy po přidání (vyznačené) operace TCG_PreconditionBefore. Návrhy jsou získané na základě obsahu sekvenčního diagramu (okno vlevo) pomocí prohledání databáze pro relevantní asociační pravidla. Uživatel může tyto návrhy využít jednoduchým přetažením vybraného kroku na libovolné místo v diagramu a nápověda se automaticky aktualizuje.

3 Vyhodnocení řešení

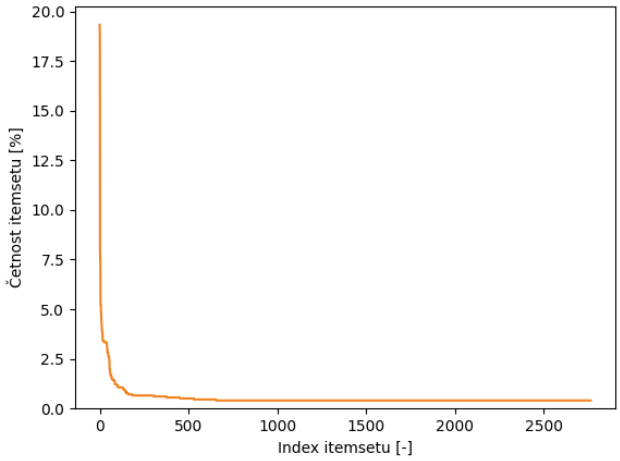
Řešení zadaného problému bylo uskutečněno na základě vývojového EXAM modelu. V této kapitole se porovnají výsledky založené na tomto modelu s jedním z produkčních modelů, využívaným zákazníkem firmy MicroNova AG.

Protože produkční model běží na nižší verzi EXAMu, než je verze aplikace s vývojovým modelem, neobsahuje pluginy, na kterých je zde popsaný plugin závislý (viz kapitolu 2.3). Není tedy v tuto chvíli možné spustit vytvořený plugin v produkčním modelu. Je ale možné z něj získat základní data (viz kapitolu 2.1) a z nich vytvořit pravidla (viz kapitolu 2.2), která by se v pluginu použila. Díky tomu je možné vykreslit grafy vlastností itemsetů a pravidel a porovnat jejich průběhy s těmi dosaženými v rámci vývojového modelu.

Dále je vyobrazen pár příkladů těchto grafů s popisem jejich porovnání s již uvedenými grafy z kapitoly 2.2.

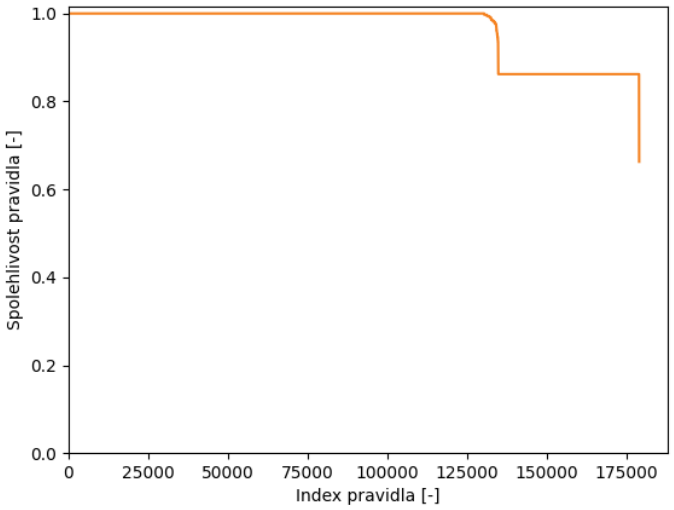


Obrázek 19: Graf seřazené četnosti itemsetů z produkčního EXAM modelu

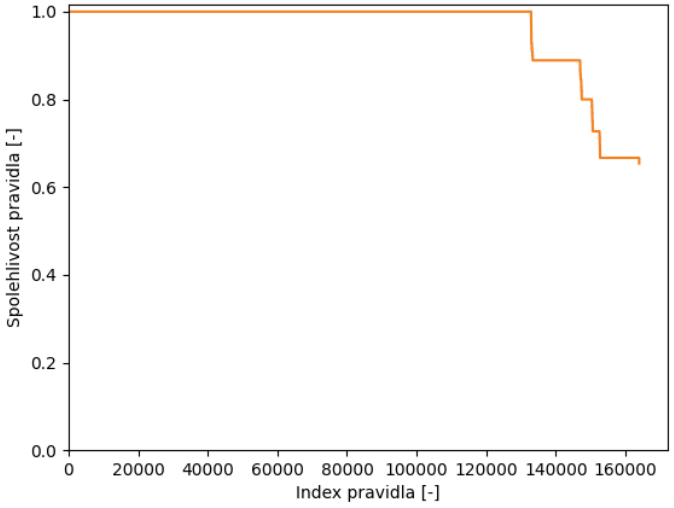


Obrázek 9: Graf seřazené četnosti itemsetů z vývojového EXAM modelu

Po porovnání grafu četnosti itemsetů získaných z produkčního modelu a jeho protějšku z vývojového modelu vidíme, že sledují téměř shodný tvar křivky s tím rozdílem, že četnost dosahuje jiného maxima a minima.



Obrázek 20: Spolehlivosti pravidel pro produkční EXAM model



Obrázek 11: Spolehlivost pravidel pro vývojový EXAM model

Z výše uvedeného grafu je zřejmé, že v produkčním modelu jsou v porovnání s vývojovým větší množiny pravidel se stejným ohodnocením spolehlivosti. Tvar grafu zůstává relativně stejný.

Z ostatních porovnání grafů je taktéž zjevná jejich podobnost. Díky tomu lze usoudit, že by se v produkčním modelu plugin choval obdobně jako ve vývojovém.

4 Závěrečné shrnutí

Tato práce se zabývala návrhem softwarového řešení pro napovídání relevantních elementů do sekvenčních diagramů v softwaru EXAM pomocí data mining algoritmů. Jejím cílem bylo vytvořit plugin rozšiřující aplikaci EXAM, který bude automaticky provádět nápovědu a umožní uživateli navržené prvky intuitivně využít.

Návrh řešení je založen na získaní dat z daného EXAM modelu pomocí Groovy skriptu v modelu samotném. Získaná data byla dále zpracována pomocí Python skriptu k vytvoření asociačních pravidel, na kterých je nápověda založena. Tato pravidla byla nově vytvořeným pluginem nahrána do jím založené SQL databáze.

Vytvořený plugin sleduje stav vyvíjeného sekvenčního diagramu a při jeho změně automaticky napovídá relevantní prvky do nového okna v aplikaci. Okno lze přesouvat, měnit jeho velikost, rolovat nabízenými prvky a jednoduše prvky využít jejich přetažením na libovolné místo v diagramu. Při míření na určitý prvek se zobrazují dodatečné informace o něm. Nabízení prvků probíhá pomocí hledání a filtrování relevantních asociačních pravidel v databázi, ke které se přistupuje pomocí JDBC rozhraní.

Cílem této práce bylo vytvořit softwarové řešení jako proof of concept, čehož bylo dosaženo. Výsledkem ověření konceptu je pak pozitivní zhodnocení vybraného přístupu k poskytování automatické nápovědy. Z toho vyplývá, že je na tomto konceptu možné a relevantní dál stavět a pokračovat tímto směrem k vývoji oficiálně podporované funkcionality v aplikaci EXAM.

Dále zpracovanými tématy by mohlo být například zefektivnění získání základních dat z EXAM modelu, zefektivnění prohledávání databáze nebo případné zredukování množiny itemsetů o ty nejvyužívanější. Pár nejčastěji využívaných kroků by mohlo být v nápovědě zobrazováno permanentně a částečně odděleně od těch dynamicky navržených.

Citovaná literatura

[1] **Crolla, David A.** *Automotive Engineering: Powertrain, Chassis Systems and Vehicle Body.* Oxford : Elsevier, 2009. ISBN 978-1-85617-577-7.

[2] **MicroNova.** EXAM Product Brochure. *micronova.de.* [Online] [Citace: 11. 12. 2021.] https://www.micronova.de/fileadmin/user_upload/Bereich_Testing_Solutions/Downloads/EXAM-Product-Brochure-EN.pdf.

[3] **Hahn, Erin N.** Johns Hopkins Applied Physics Laboratory. *Techdigest.* [Online] Volume 32, 2014. [Citace: 4. 1. 2022.] https://www.jhuapl.edu/Content/techdigest/pdf/V32-N04/32-04-Hahn.pdf.

[4] **MicroNova.** EXAM Testautomation. *micronova.de.* [Online] [Citace: 11. 12. 2021.] Dostupné z: https://www.micronova.de/en/testing/exam-testautomation.html.

[5] **National Instruments.** What Is Hardware in the Loop? *ni.com.* [Online] National Instruments. [Citace: 11. 1. 2022.] https://www.ni.com/cs-cz/innovations/white-papers/17/what-is-hardware-in-the-loop-.html.

[6] **MicroNova.** NovaCarts Engine. *micronova.de.* [Online] MicroNova. [Citace: 11. 1. 2022.] https://www.micronova.de/en/testing/novacarts/hil-simulators/novacarts-engine.html.

[7] **OPAL-RT.** opal-rt.com. *software in the loop.* [Online] OPAL-RT. [Citace: 14. 1. 2022.] https://www.opal-rt.com/software-in-the-loop/.

[8] **Hamilton, Kim a Miles, Russell.** *Learning UML 2.0.* Sebastopol : O'Reilly Media, Inc., 2006. ISBN: 0-596-00982-8.

[9] *Test Sequence Generation from UML Sequence Diagrams.* **Samuel, Philip a Joseph, Anju Teresa.** Phuket : IEEE, 2008. DOI: 10.1109/SNPD.2008.100.

[10] **McAffer, Jeff, Lemieux, Jean-Michel a Aniszczyk, Chris.** *Eclipse Rich Client Platform.* Second Edition. Boston : Pearson Education, Inc., 2010. ISBN 0321603788.

[11] **Oracle.** The Java Language Environment. *oracle.com.* [Online] [Citace: 6. 4 2022.] https://www.oracle.com/java/technologies/introduction-to-java.html#334.

[12] **Horstmann, Cay S.** *Core Java.* Twelfth Edition. Boston : Addison-Wesley Professional, 2021. Vol. I: Fundamentals. ISBN: 0137673620.

[13] **König, Dierk, a další.** *Groovy in action.* Second Edition. Shelter Island : Manning Publications Co., 2015. ISBN: 9781935182443.

[14] **Pecinovský, Rudolf.** *Python Kompletní příručka pro verzi 3.9.* Praha : Grada Publishing, 2020. ISBN: 9788027112692.

[15] **Beaulieu, Alan.** *Learning SQL.* Second Edition. Sebastopol : O'Reilly Media, Inc., 2009. ISBN-13: 978-0-596-52083-0.

[16] **Date, C. J.** *Database Desing and Relational Theory.* Second Edition. Healdsburg : Apress, 2019. ISBN: 1484255399.

[17] **Zhang, Chengqi a Zhang, Shichao.** *Association Rule Mining Models and Algorithms.* New York : Springer-Verlag Berlin Heidelberg, 2002. ISBN: 3-540-43533-6.

[18] **pandas.** About pandas. *pandas.org.* [Online] NumFOCUS. [Citace: 1. 5. 2022.] https://pandas.pydata.org/about/.

[19] **The pandas development team.** pandas-dev/pandas: Pandas 1.4.2. *zenodo.org.* [Online] Zenodo, 2. 4. 2022. [Citace: 1. 5. 2022.] https://doi.org/10.5281/zenodo.6408044. DOI: 10.5281/zenodo.6408044.

[20] *MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack.* **Raschka, Sebastian.** 24, joss.theoj.org : The Open Journal, 2018, Journal of Open Source Software, Sv. 3. DOI: 10.21105/joss.00638.

[21] **Raschka, Sebastian.** association_rules: Association rules generation from frequent itemsets. *mlxtend.* [Online] 2020. [Citace: 3. 5. 2022.] https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/.

[22] **H2.** H2 Database Engine. *h2database.com.* [Online] [Citace: 5. 5. 2022.] https://www.h2database.com/html/main.html.

Seznam obrázků, tabulek a rovnic

Obsah přiloženého CD

- Text bakalářské práce – formát .pdf
- Zdrojový kód Groovy skriptu – formát .groovy
- Zdrojový kód Python skriptu – formát .py
- Repositář s obsahem Eclipse pluginu

Příloha A: Hlavní okno aplikace EXAM s otevřeným sekvenčním diagramem