Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Uživatelské rozhraní prezentující obsah datového skladu nástroje SPADe

Plzeň, 2017 Michal Všelko

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal VŠELKO**
Osobní číslo:	**A14B0073K**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Název tématu:	**Uživatelské rozhraní prezentující obsah datového skladu nástroje SPADe**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování:

1. Prozkoumejte strukturu metamodelu SPADe a možnosti technologií pro vykreslování grafů v desktopových i webových aplikacích.
2. Analýzou dat v datovém skladu SPADe vyberte vhodnou množinu statistických údajů pro zobrazení (průměr, suma, histogram, atd.) a možnosti filtrace výchozích dat.
3. S pomocí zvolené technologie implementujte aplikaci umožňující zobrazení dat v datovém skladu SPADe a jejich charakteristik odpovídajících předchozímu bodu zadání ve formě vhodně navržených grafů.
4. Výslednou aplikaci otestujte z uživatelského i funkčního pohledu.

Rozsah grafických prací:	**dle potřeby**
Rozsah kvalifikační práce:	**doporuč. 30 s. původního textu**
Forma zpracování bakalářské práce:	**tištěná**
Seznam odborné literatury:	
dodá vedoucí bakalářské práce	

Vedoucí bakalářské práce:	**Ing. Petr Pícha** Katedra informatiky a výpočetní techniky
Datum zadání bakalářské práce:	**10. října 2016**
Termín odevzdání bakalářské práce:	**4. května 2017**

Doc. RNDr. Miroslav Lávička, Ph.D.
děkan

L.S.

Doc. Ing. Přemysl Brada, MSc. PhD.
vedoucí katedry

V Plzni dne 13. října 2016

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 27.6.2017 Michal Všelko**,**

Abstract

The goal of this thesis is to create a graphical interface for the SPADe data warehouse. The graphical interface displays statistics and charts of the data stored in the SPADe data warehouse. The thesis includes a description of basic types of charts and comparison of libraries for their plotting.

The benefit of this thesis is the creation of a program that will improve and facilitate identification and presentation of frequent bad practices in software development in the future. The software will thus help to improve project management processes, which is the main goal of the overall SPADe project.

Abstrakt

Cílem této práce je vytvořit grafické rozhraní k datovému skladu nástroje SPADe. Grafické rozhraní zobrazuje statistiky a grafy dat uložených v datovém skladu nástroje SPADe. Součástí práce je popis základních typů grafů a srovnání knihoven pro jejich vykreslení.

Přínosem práce je vytvoření programu, který díky svému uživatelskému rozhraní v budoucnu zlepší a usnadní identifikaci a prezentaci často opakovaných chyb při vývoji software a napomůže tak ke zlepšení procesů řízení projektů, což je hlavním cílem celého projektu SPADe.

Obsah

1 Úvod

V současné době je na Katedře informatiky a výpočetní techniky (KIV) Fakulty aplikovaných věd (FAV) Západočeské univerzity v Plzni (ZČU) vyvíjen nástroj Software Process Anti-patterns Detector (SPADe) pro vyhledání takzvaných anti-patternů (často opakovaných chyb) v průběhu projektů vývoje software na základě analýzy projektových dat z Application Lifecycle Management (ALM) nástrojů. Data tohoto nástroje se ukládají do datového skladu.

Cílem bakalářské práce je vytvořit grafické rozhraní pro zobrazení obsahu tohoto datového skladu. Pro výběr konkrétních údajů je nutné provést analýzu datového skladu a vybrat relevantní data k zobrazení. Grafické rozhraní bude prezentovat uživateli vybrané statistické údaje pomocí grafů a výpisů informací. Práce obsahuje popis nejčastěji používaných typů grafů a srovnání knihoven pro jejich vykreslení v desktopové i webové aplikaci. Z důvodu velkého množství dat je nutné vytvořit v aplikaci možnost filtrování zobrazených informací podle různých kritérií.

Součástí práce je krátký popis nástroje SPADe. Následuje popis jednotlivých základních typů grafů, u kterých je předpoklad pozdějšího využití při implementaci nebo v budoucím rozvoji grafického rozhraní. Na tento popis navazuje část práce, zabývající se srovnáním knihoven sloužících pro vykreslení jednotlivých grafů v desktopové či webové aplikaci. Dále práce obsahuje analýzu datového skladu za účelem návrhu datového modelu výsledné aplikace, návrh grafického rozhraní aplikace, popis její implementace a průběhu jejího testování.

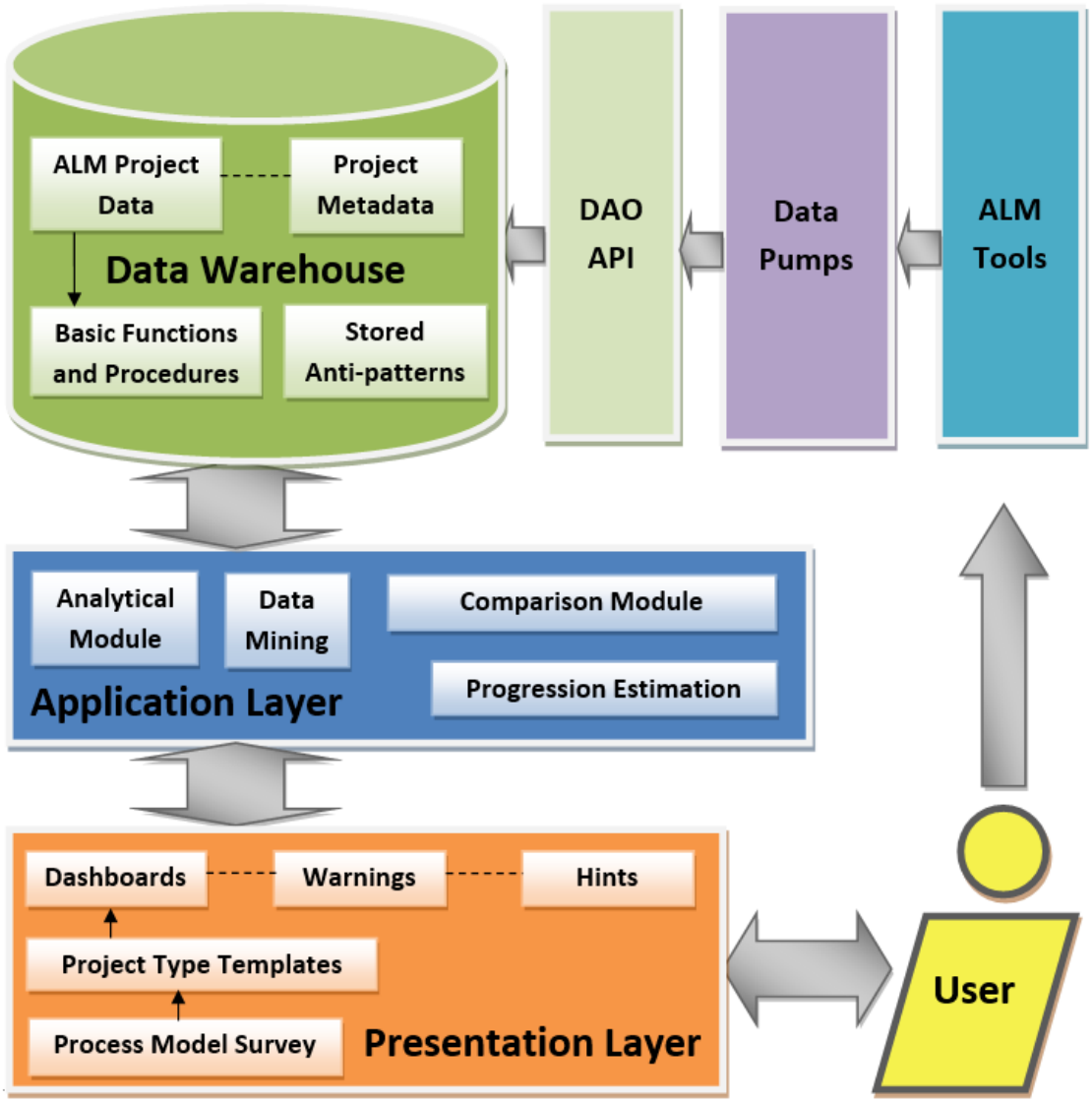
2 Software Process Anti-patterns Detector

SPADe (Software Process Anti-patterns Detector) [1] [2] [3] je softwarový nástroj vyvíjený na Katedře informatiky a výpočetní techniky (KIV) Fakulty aplikovaných věd (FAV) Západočeské univerzity v Plzni (ZČU). Tento nástroj doluje data pomocí datových pump z jednotlivých Application Lifecycle Management (ALM) nástrojů.

Datová pumpa je jednoduchý ETL (Extract – Transform – Load) software, který vytěžuje data z daného ALM nástroje a mění jejich strukturu do požadovaného cílového formátu. Získaná data následně ukládá v tomto případě do datového skladu nástroje SPADe.

ALM nástroje slouží k řízení životního cyklu aplikace v jednom nebo více stádiích. Slouží například ke sledování změn v projektu, správě požadavků, ukládání verzí produktu atd. Mezi ALM nástroje patří například VCS (Version Control System) nástroje jako Apache Subversion (SVN) či Git, CCM (Configuration and Change Management) nástroje pro řízení a sledování změn (také nazývané issue-tracking nástroje) jako Github, Assembla, Bugzilla, Atlassian Jira a jiné.

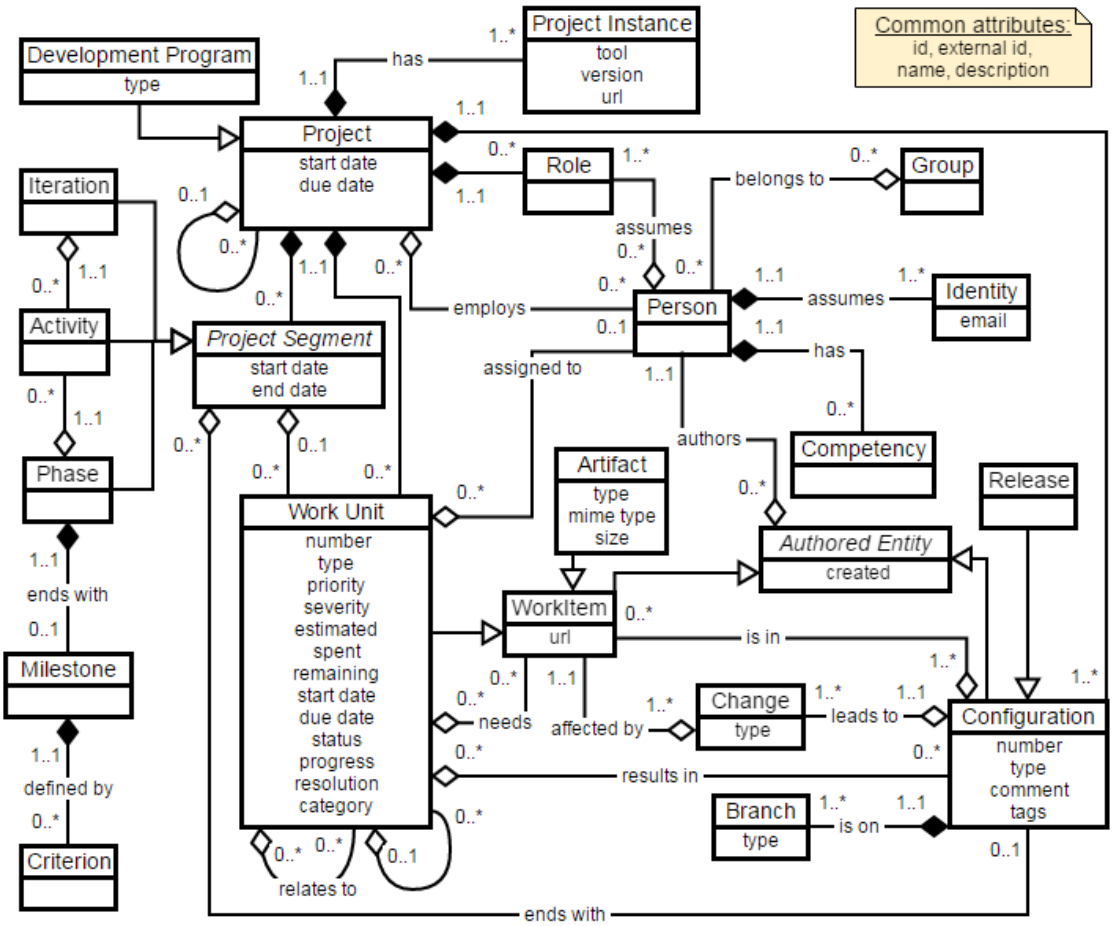
SPADe data ukládá do datového skladu a následně identifikuje často opakované chyby managementu nebo vývojových týmů v projektech. Tyto chyby se nazývají anti-patterny nebo bad practices. Schéma nástroje SPADe je na obrázku 1.



Obrázek 1: Schéma nástroje SPADe [1]

Datový sklad (Data Warehouse) je úložiště velkého množství dat určené k analytickému dotazování. Pojem Data Warehouse (DWH) definoval William H. Inmon jako „*subjektově orientovanou, integrovanou, časově proměnnou a stálou kolekci dat pro podporu rozhodování managmentu*“ [4]. Datový sklad je tedy organizován kolem subjektů (např. zákazník, produkt …), čímž je jeho struktura čitelnější pro uživatele. Integrovanost datového skladu spočívá v ukládání dat z různých zdrojů a jejich následného seskupení podle logického významu. Data se ukládají za delší časový horizont než je tomu u relační databáze, kde je důležitější aktuální stav dat. Stálost datového skladu znamená, že se obvykle po nahrání data již neupravují. Doménový model nástroje SPADe ukazuje obrázek 2.

Podrobný a k datu odevzdání aktuální fyzický model je pak v příloze D a také na kompaktním disku přiloženém k této práci. Podrobnější informace o celém SPADe lze najít v [1] [2] [3].



Obrázek 2: Doménový model nástroje SPADe [2]

Pro rozšíření funkčnosti nástroje SPADe je nutné vytvořit webovou či desktopovou aplikaci s grafickým uživatelským prostředím (Graphical User Interface, GUI), která bude zobrazovat statistiky a grafy popisující obsah datového skladu a dá uživateli možnost filtrovat zobrazení údajů podle potřeby. Taková aplikace je výstupem této práce.

3 Grafy

Pojem graf lze interpretovat několika možnými způsoby. Graf si lze vyložit v kontextu teorie grafů jako soustavu vrcholů, které jsou vzájemně spojeny orientovanými či neorientovanými hranami (v angličtině graph). Takové grafy ovšem nejsou součástí sledované práce. Pod pojmem graf v kontextu této práce uvažujeme grafické znázornění vhodné pro reprezentaci numerických hodnot (např. množství, sum, průměrů, rozložení hodnot atd.) dat z datového skladu nástroje SPADe (v angličtině chart).

Grafy [5] mají za úkol vizuálně reprezentovat informace uživateli. Musí z nich být patrné, jaká data a hodnoty zobrazují. Každý graf by měl mít svůj název. U většiny grafů je nedílnou součástí také legenda zobrazených hodnot. Některé grafy je možné zobrazit v trojrozměrném prostoru. Toto zobrazení ale snižuje čitelnost a přehlednost. Grafy ve 3D tudíž nejsou součástí této práce.

Existuje několik základních typů grafů, které byly vyhodnoceny jako vhodné pro reprezentaci dat DWH nástroje SPADe.

3.1 Osové grafy

Tento typ grafů disponuje osami hodnot. Obvykle obsahuje dvě; osu X a Y (při zobrazení 3D grafu je navíc osa Z). Speciálním typem jsou časové grafy, u nichž (převážně) osa X obsahuje časovou řadu. Na ose Y jsou poté hodnoty naměřené v těchto časových intervalech nebo jiným způsobem spjaty s patřičnými časovými údaji. Pomocí těchto grafů je tedy možné sledovat vývoj dat v čase.

3.1.1 Bodový graf

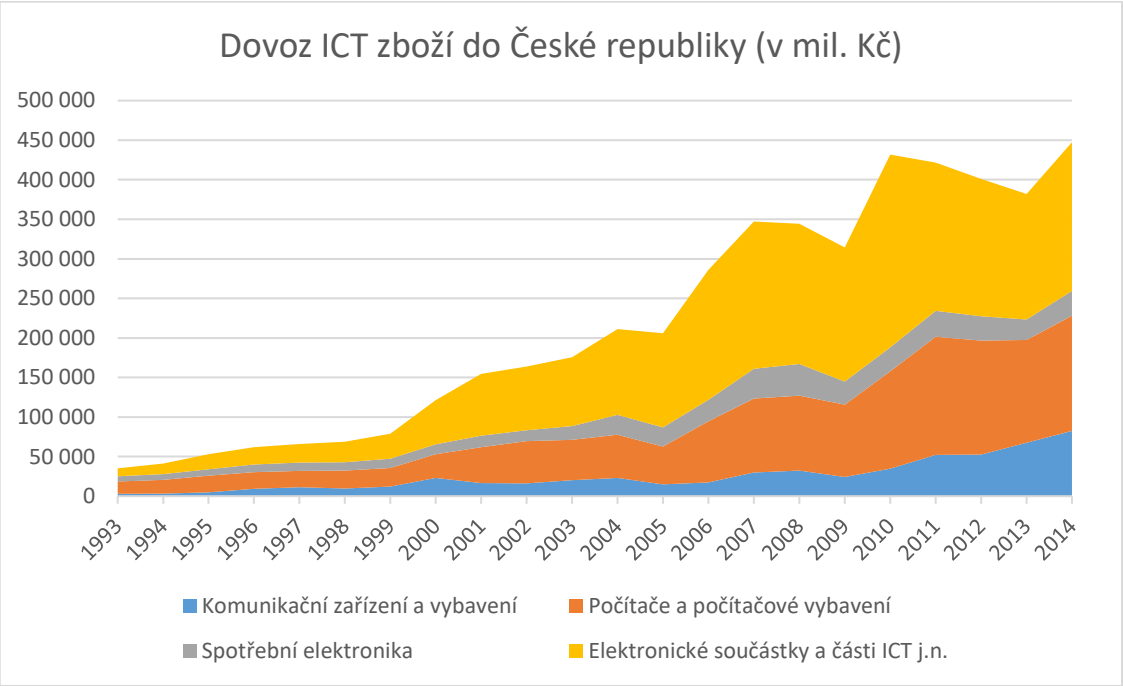
Bodový graf zobrazuje vztah mezi dvěma proměnnými v souřadnicovém systému. Zobrazuje body reprezentující jednotlivé souřadnice proměnné, x a y. Pro zobrazení dalších dimenzí je možné využít různých velikostí, barev a tvarů bodů (při zachování 2D zobrazení). Tento způsob ovšem značně snižuje čitelnost grafu. Je vhodný pro zobrazení hustoty bodů, která by nebyla patrná ze spojnicového grafu (viz níže).

3.1.2 Spojnicový graf

Je podobný bodovému grafu. Jednotlivé body kartézského součinu (tj. souřadnic bodů na osách X a Y) jsou ale spojeny čarami nebo křivkami. Spojnicového grafu se využívá pro zobrazení trendů; jeho zástupcem je např. zobrazení matematické funkce.

3.1.3 Plošný graf

Jedná se o obdobu spojnicového grafu. Rozdíl oproti spojnicovému grafu spočívá v tom, že plocha mezi spojnicí bodů a osou x je zabarvená. Rozlišují se dva typy plošného grafu – jednoduchý a skládaný. Zabarvení jednoduchého grafu je poloprůhledné, čímž hodnota začíná vždy na nule. Oproti tomu skládaný graf poloprůhledný není a hodnota tak začíná na horizontu předešlé spojnice. Příklad skládaného plošného grafu je zobrazen na obrázku 3.

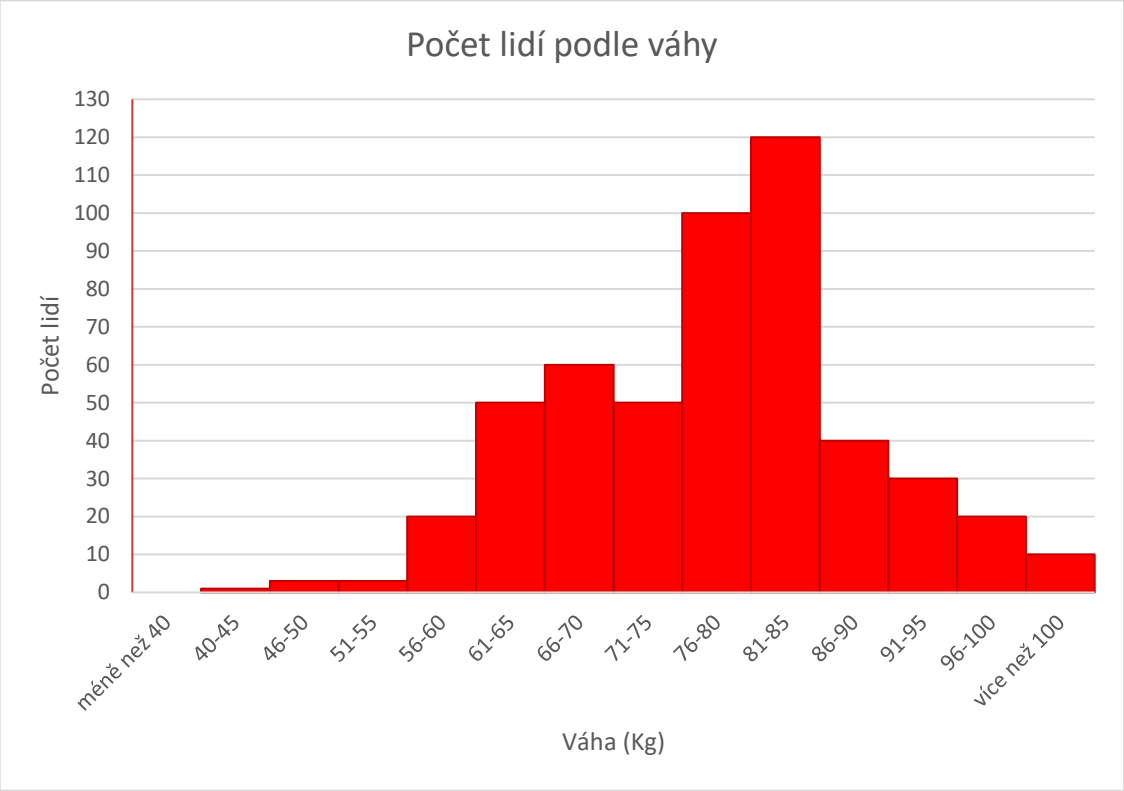


Obrázek 3: Plošný graf

3.1.4 Sloupcový graf

Tento typ grafu zobrazuje obdélníky s výškou či délkou odpovídající velikosti hodnoty. Sloupce lze použít svislé či vodorovné. Vodorovný typ grafu se využívá v případě dlouhých popisků. Častější a přehlednější je sloupcový graf se svislými sloupci.

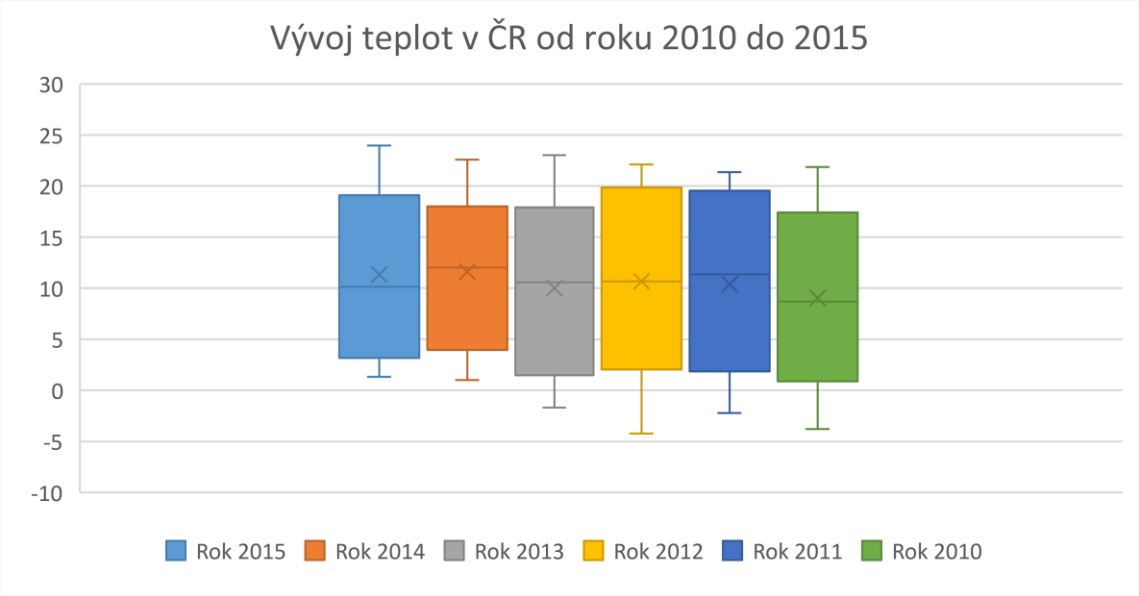
Zvláštním typem sloupcového grafu je histogram, který zobrazuje počet prvků v určitém intervalu. Prvky jsou shlukovány do jednotlivých tříd podle určitých kritérií. Tyto třídy určují osu X a četnost prvků v každé třídě určuje osu Y. Tím je graficky znázorněn trend velikosti veličiny. Sloupce histogramu mají stejnou šířku. Příklad histogramu je zobrazen na obrázku 4.



Obrázek 4: Histogram

3.1.5 Tukey box graf

Využívá se pro zobrazení statistických parametrů při opakovaném měření hodnot. V Tukey boxu se následně zobrazí průměrná, minimální a maximální hodnota, 25%, 50% a 75% kvantil jedné proměnné. Ukázka Tukey box grafu je na obrázku číslo 5.



Obrázek 5: Tukey box graf

3.2 Kruhové grafy

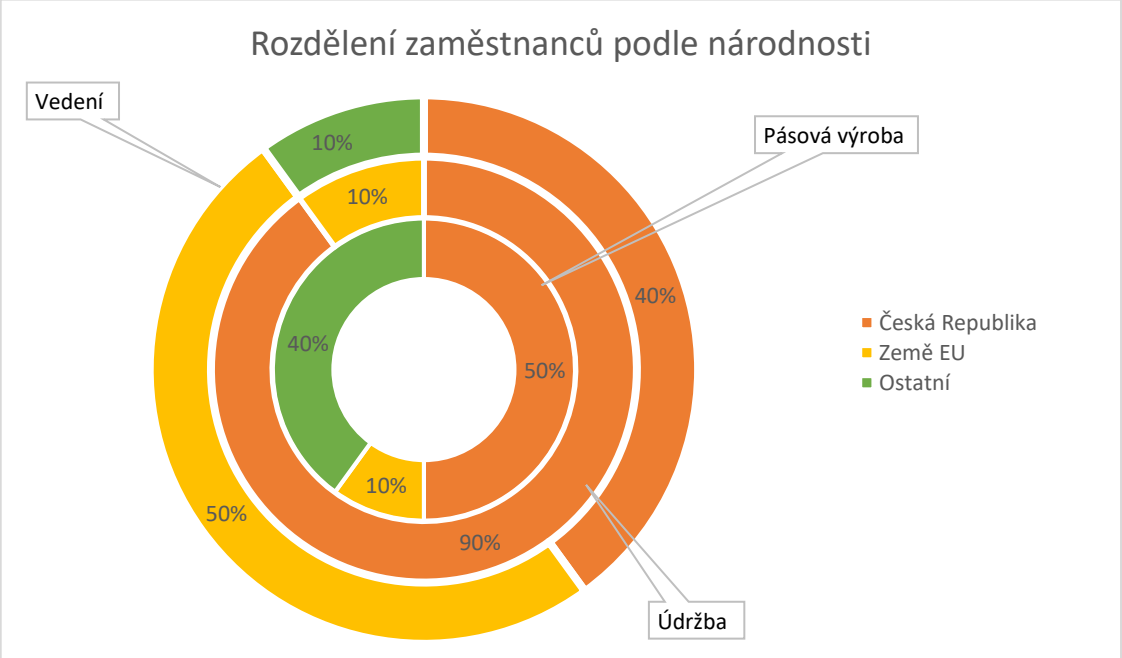
Tento typ grafů zobrazuje procentuální zastoupení hodnoty z celku. Oproti osovým grafům neobsahují kruhové grafy osy, nelze zobrazit vzájemnou závislost dvou různých hodnot. Kruhové grafy také nejsou vhodné pro zobrazení časové řady. Pro přehlednost je vhodný menší počet zobrazovaných částí. V opačném případě je možné kruhový graf nahradit některým z osových grafů, například sloupcovým.

3.2.1 Výsečový graf

Výsečový graf zobrazuje částečné zastoupení hodnoty z celkové velikosti. Graf má tvar kruhu a každá hodnota určuje velikost výseku z celku. Při větším počtu hodnot se graf stává nepřehledným a je lepší použít sloupcový graf.

3.2.2 Prstencový graf

Rozdíl tohoto grafu oproti výsečovému je v jeho tvaru. Výsečový graf má tvar kruhu, zatímco prstencový graf má tvar prstence. Tím je umožněno zobrazení více datových řad. Každá datová řada má svůj vlastní prstenec, nazývající se úroveň. Prstence jsou do sebe zapuštěny; vzniklý graf se nazývá víceúrovňový. Ukázka víceúrovňového prstencového grafu je na obrázku 6.



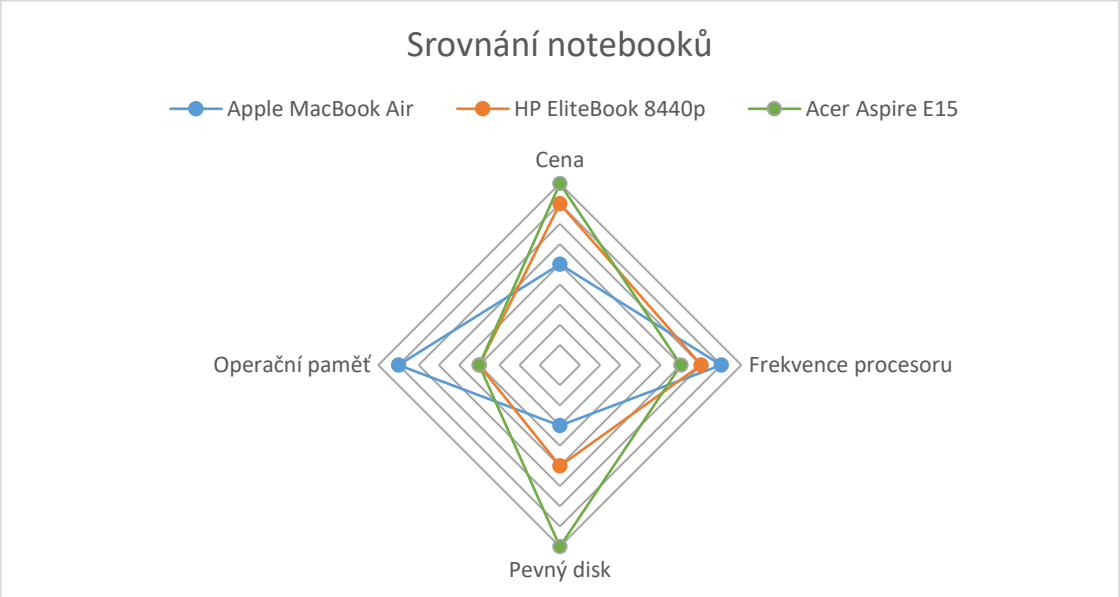
Obrázek 6: Prstencový graf

3.3 Ostatní grafy

Mezi ostatní grafy byly zařazeny takové typy, které neodpovídají definici osových nebo kruhových grafů. Do uvedené skupiny lze zařadit například různé budíky či teploměry, mapy nebo Sankeyův diagram. Zmíněné grafy ovšem zřejmě nenajdou využití pro zobrazení dat nástroje SPADe, i když v současné chvíli nelze vyloučit, že se některý z těchto typů grafů v budoucí podobě grafického rozhraní objeví. Jako reprezentanti tohoto typu byli vybráni pavučinový graf a Ganttův diagram, které jsou také specifické a existuje u nich možnost pozdějšího využití v uživatelském rozhraní nástroje SPADe.

3.3.1 Pavučinový graf

Tento typ by spadal do skupiny grafů, která je mezi osovými a kruhovými. Svým tvarem by se dal řadit do kruhových grafů, ale reprezentací hodnot by spadal spíše do osových. Graf totiž nezobrazuje procentuální zastoupení hodnoty z celku, jako je tomu u kruhových grafů, ale z jeho středu vychází paprsky představující stupnici jednotlivých hodnot. Hodnoty vyznačené na paprscích jsou spojeny úsečkami. Tím v podstatě vzniká spojnicový graf pro více typů hodnot. Ukázka pavučinového grafu je na obrázku 7.

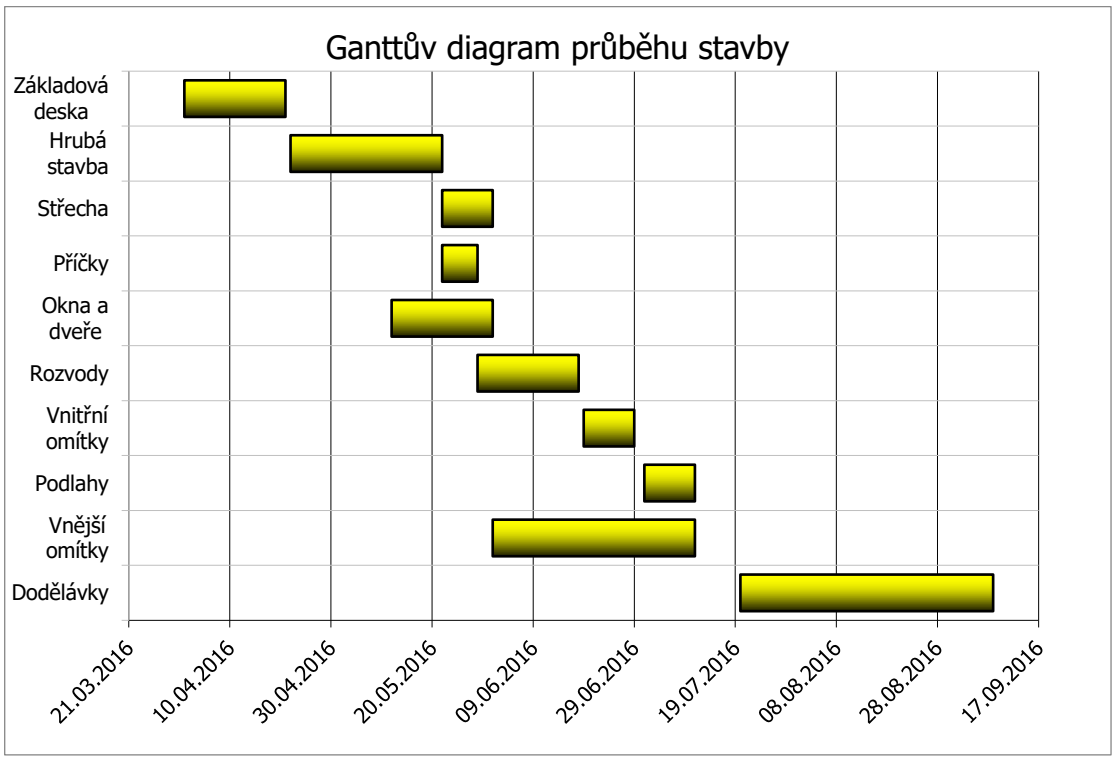


Obrázek 7: Pavučinový graf

3.3.2 Ganttův diagram

Ganttův diagram znázorňuje průběh, časovou návaznost a paralelizaci činností v časovém rozmezí. Na ose Y jsou vypsány jednotlivé činnosti. Osa X slouží pro zobrazení časového intervalu. Jednotlivé velikosti horizontálních sloupců určují dobu strávenou danou

činností, popř. pro danou činnost plánovanou. Příklad Ganttova diagramu ukazuje obrázek 8.



Obrázek 8: Ganttův diagram

4 Knihovny pro vykreslení grafů

Pro vykreslení grafů jak v desktopové, tak webové aplikaci existuje celá řada knihoven. Zvláště knihoven pro desktopovou aplikaci je velké množství. V bakalářské práci byly porovnávány knihovny, které byly často doporučovány v článcích, knihách a internetových fórech zabývajících se tímto tématem [6] [7] [8] [9] [10].

Knihovna pro tuto práci musí splňovat určitá kritéria. Důležitým kritériem je licence. Pro srovnání byly vybrány knihovny, jejichž distribuce je pod některou z freeware (volně dostupných) licencí jako LGPL (Library General Public License) nebo BSD (Berkeley Software Distribution). Dalším kritériem je implementace základních typů grafů z kapitoly 3 a možnosti nastavení vizuální stránky grafů (např. barvy, měřítka, zobrazované hodnoty na osách a jejich formát atd.).

Pro výběr knihovny hraje svou roli i programovací jazyk. Samotný nástroj SPADe je implementován v jazyce Java a je tedy žádoucí, aby i grafické rozhraní bylo implementováno stejnou technologií. Zadáním nebylo stanoveno, zda má být knihovna desktopová či webová, tudíž i kdyby webová knihovna nebyla implementována přímo v Javě, musí s ní alespoň komunikovat.

4.1 Desktop

4.1.1 Gral

Gral (GRAphing Library) [5] [11] je knihovna pod licencí LGPL. Nabízí vykreslení většiny základních typů grafů. S její pomocí lze zobrazit bodový, spojnicový, plošný, sloupcový, výsečový, prstencový a Tukey box graf. Lze jej vyexportovat do formátů PNG (Portable Network Graphics), GIF (Graphics Interchange Format), JPEG (Joint Photographic Experts Group), EPS (Encapsulated PostScript), PDF (Portable Document Format) a SVG (Scalable Vector Graphics) nebo odeslat přímo k tisku. Knihovna nabízí velké množství nastavení a stylizování.

4.1.2 JCCKit

JCCKit (Java Chart Construction Kit) [5] [12] je starší knihovna distribuována pod licencí LGPL. Její poslední verze byla vydána v lednu 2005. Výhodu představuje velikost samotného souboru jcckit.jar, který má pouze 97 kB. Nevýhoda je naproti tomu množství

typů grafů, které knihovna nabízí. Lze využít pouze nejzákladnější typy, jako jsou bodové, spojnicové a sloupcové grafy.

4.1.3 JFreeChart

JFreeChart [13] [14] [15] je nejpoužívanější a nejkomplexnější knihovna pro zobrazení grafů. Je založena na Java2D API (Application Programming Interface). Knihovna zvládne bodový, spojnicový, plošný, sloupcový, výsečový, prstencový, pavučinový i Tukey box graf. Zajímavou nadstavbou knihovny je možnost vytvoření Ganttova diagramu a dalších zvláštních typů grafů, jako jsou různé budíky a teploměry. Některé z těchto typů lze zobrazit ve 3D (čárový, sloupcový, výsečový). JFreeChart nabízí i kombinování více typů grafů do jednoho. Knihovna disponuje celou řadou možností nastavení a stylizování grafů. Výsledný graf lze snadno odeslat do tisku nebo exportovat do formátů PNG, SVG a PDF pomocí vyvolání kontextového menu pravým tlačítkem myši. Knihovna je distribuována pod licencí LGPL.

4.1.4 jCharts

jCharts [5] [16] je knihovna, jejíž poslední verze byla vydána v roce 2004. Knihovna umí vykreslit bodový, spojnicový, plošný, sloupcový, koláčový výsečový a Tukey box graf. Jednotlivé typy lze kombinovat do jednoho grafu. Knihovna je distribuována pod licencí BSD.

4.1.5 OpenChart2

Knihovna OpenChart2 [5] [17] navazuje na vývoj knihovny JOpenChart ukončený v roce 2002. Vývoj knihovny OpenChart2 byl ukončen v roce 2009. Knihovna nabízí vykreslení bodového, spojnicového, sloupcového, výsečového a pavučinového grafu. S předchozími knihovnami má společnou distribuci pod licencí LGPL.

4.2 Web

4.2.1 Google Charts

Google Charts [18] [19] je nástroj pro online tvorbu grafů od společnosti Google. Ze základních typů grafů lze pomocí něj vykreslit na webových stránkách bodový, spojnicový, plošný, sloupcový, výsečový, prstencový a Tukey box graf. Dále nabízí také graf pro zobrazení organizační struktury, Ganttův diagram nebo budík. Google Charts také nabízí kombinování více grafů do jednoho. Graf se ovládá pomocí argumentů v URL

(Uniform Resource Locator) adrese. Následně se zobrazí na stránce jako PNG obrázek. Díky tomu není třeba licence.

4.2.2 Chart.js

Chart.js [19] [20] je javascriptová open-source knihovna. Její výhodou je jednoduchost, díky čemuž je velice populární. Pomocí této knihovny lze zobrazit bodový, spojnicový, sloupcový, výsečový, prstencový a pavučinový graf. Všechny grafy jsou responzivní [1].

4.2.3 Chartist.js

Chartist.js [19] [21] je javascriptová knihovna. Vytváří SVG grafy, které lze upravovat pomocí CSS (Cascading Style Sheets) stylů. Grafy jsou responzivní. Knihovna umí zobrazit bodový, spojnicový, plošný, sloupcový, výsečový a prstencový graf.

4.2.4 pChart 2.0

pChart 2.0 [22] je PHP (Hypertext Preprocessor, Personal Home Page) knihovna určená pro vytváření grafů na straně serveru. Nabízí velikou škálu typů grafů k zobrazení. Ze základních typů grafů umí vykreslit všechny. Tato knihovna je šířena pod licencí GNU GPL (GNU's Not Unix General Public License). Pro komerční použití je nutné licenci zakoupit.

4.2.5 JpGraph

Další z řady knihoven pro vykreslení grafů pomocí PHP [23]. Tato knihovna umí vykreslit bodový, spojnicový, plošný, sloupcový, výsečový a pavučinový graf.

4.3 Srovnání

Kritérii pro srovnání vybraných knihoven byla schopnost zobrazit co největší množství základních typů grafů, i když pravděpodobně nebudou všechny použity. Díky tomu budou zachovány široké možnosti výběru grafu pro konkrétní hodnoty při samotné implementaci. Dalšími kritérii pro srovnání jsou možnosti vizuálního nastavení, licence (či obecně legální možnost pořízení) zdarma a v neposlední řadě také dokumentace knihoven. Kritéria jsou stejná pro knihovny webových i desktopových aplikací. Srovnání jednotlivých knihoven shrnuje tabulka 1. V tabulce se neuvádí, zda knihovna umí zobrazit graf ve 3D, protože tyto typy grafů nebudou součástí grafického rozhraní nástroje SPADe.

[1] Responzivnost znamená, že zobrazení grafu je automaticky optimalizováno pro maximální přehlednost a informační hodnotu při zobrazení v zařízeních různých velikostí, při změně velikosti okna, přiblížení nebo oddálení

Vizuální možnosti knihoven nebylo potřeba do tabulky uvádět. Všechny knihovny v podstatě splňovaly možnosti základního nastavení (např. barvy, legenda, měřítka atd.).

Ve sloupci pro dokumentaci se nehodnotila pouze uživatelská dokumentace dodávaná ke knihovnám, ale také množství příkladů a řešení obtížných situací na různých stránkách a diskuzních fórech. Toto kritérium se hodnotilo třemi úrovněmi. Nejlépe popsané knihovny byly hodnoceny jako obsáhlé. Knihovny s popisem splňující nároky pro úspěšnou tvorbu grafu začátečníkem byly hodnoceny jako dostatečné. Nejhorší úroveň dokumentace (nedostatečná) nebyla využita.

	Základní typy grafů								Licence	Dokumentace
	Bodový	Spojnicový	Plošný	Sloupcový	Pavučinový / Gantt	Tukey box	Výsečový	Prstencový		
Gral	✓	✓	✓	✓	✗ / ✗	✓	✓	✓	GNU LGPL	Dostatečná
JCCKit	✓	✓	✗	✓	✗ / ✗	✗	✗	✗	GNU LGPL	Dostatečná
JFreeChart	✓	✓	✓	✓	✓ / ✓	✓	✓	✓	GNU LGPL	Obsáhlá
jCharts	✓	✓	✓	✓	✗ / ✗	✓	✓	✗	BSD	Obsáhlá
OpenChart2	✓	✓	✗	✓	✓ / ✗	✗	✓	✗	GNU LGPL	Dostatečná
Google Charts	✓	✓	✓	✓	✗ / ✓	✓	✓	✓	-	Obsáhlá
Chart.js	✓	✓	✗	✓	✓ / ✗	✗	✓	✓	MIT	Obsáhlá
Chartist.js	✓	✓	✓	✓	✗ / ✗	✗	✓	✓	MIT, WTFPL	Obsáhlá
pChart 2.0	✓	✓	✓	✓	✓ / ✗	✓	✓	✓	GNU GPL	Dostatečná
JpGraph	✓	✓	✓	✓	✓ / ✓	✗	✓	✗	QPL	Dostatečná

Tabulka 1: Srovnávací tabulka

Ze srovnávací tabulky vyplývá, že nejlépe dopadla knihovna JFeeChart. Tato knihovna obsahuje všechny základní typy grafů, licence je šířena pod GNU LGPL a díky své popularitě je velice dobře popsána. Z pohledu zobrazení počtu základních typů grafů dopadla velmi dobře také PHP knihovna pChart 2.0, ale PHP je vzdálenější od Javy ve které je implementován nástroj SPADe a navíc je možnost použití knihovny omezeno nutností zakoupení licence v případě komerčního využití. Velmi dobře dopadl také nástroj společnosti Google, který by byl nejspíše volbou pro grafické rozhraní webové aplikace.

Knihovna JFreeChart odpovídá všem požadavkům a je tedy velmi dobrou volbou pro grafické rozhraní nástroje SPADe. S výběrem knihovny pro vykreslení grafů souvisí i výběr knihovny pro grafické zobrazení aplikace. JFreeChart je primárně určena pro knihovnu Swing. Bylo by možné použít novější JavaFX, ale v takovém případě by se musely grafy převádět do SwingNode, což je objekt sloužící pro vložení Swing objektů

do JavaFX. Swing obsahuje všechny možnosti, které budou potřeba pro vytvoření grafického rozhraní a použití JavaFX nebylo podmínkou zadání. Není tedy důvod nuceně preferovat použití JavaFX oproti Swingu.

5 Návrh grafického rozhraní

5.1 Analýza datového skladu SPADe

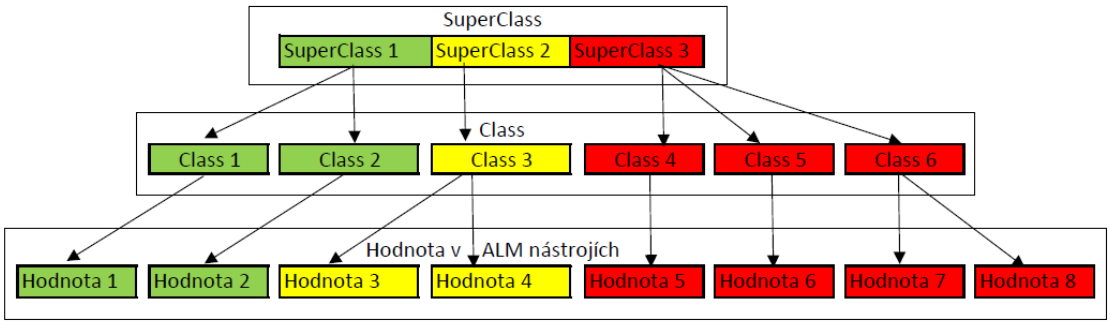
Data datového skladu nástroje SPADe jsou uložena v MySQL (My Structured Query Language) databázi. V této části práce budou popsány některé domény (viz obrázek 2), které budou využity při tvorbě grafického rozhraní. Budou-li zmíněny přímo tabulky fyzického datového modelu (příloha D), které nejsou součástí jeho doménové podoby, bude tento fakt explicitně zdůrazněn.

5.1.1 Významné doménové entity

Project

Doména *Project* obsahuje údaje o jednotlivých projektech. Každý projekt má v této doméně jeden záznam, který obsahuje základní informace jako je identifikátor, název projektu nebo počáteční a koncové datum. Identifikátorem projektu jsou k této entitě připojeny domény segmentů. Konkrétní záznamy segmentů jsou uloženy v objektech *Phase*, *Iteration* nebo *Activity*. Jeden projekt může obsahovat více fází, iterací a aktivit. Dále jsou k projektu připojeny záznamy úkolů z domény *Work Unit*. Jednotlivé úkoly mohou být přiřazeny k některému segmentu nebo přímo k projektu. Přímo k projektu jsou připojeny také domény *Configuration*, obsahující záznamy o jednotlivých konfiguracích (viz podkapitola Configuration kapitoly 5.1.1), a *Person* s údaji o osobách podílejících se na konkrétním projektu.

Přes doménu *Project Instance* (obsahující reprezentace projektů v jednotlivých ALM, kterých může – a často tomu tak je – projekt používat více) jsou ve fyzickém datovém modelu nástroje SPADe připojeny entity s hodnotami priorit, severit (závažností), statusů (stavů úkolu v jeho životním cyklu, např. „*open*“ a „*closed*“), resolucí (typu vyřešení úkolu, např. „*duplicate*“, „*invalid*“ nebo „*fixed*“) a typů úkolů (např. „*bug*“, „*feature*“ nebo „*support*“), které jsou v projektu používány. Priority, severity, statusy, resoluce a typy mohou mít různé konkrétní hodnoty (jejich počet i terminologii) v různých nástrojích, ze kterých se data dolují. Proto jsou ke každé této výčtové tabulce připojeny kvalifikační tabulky s atributy *class* a *superclass*, které údaje sdružují do skupin ve dvou hierarchických úrovních, a tím poskytují možnost posuzovat hodnoty jednotným způsobem nezávisle na zdroji dat (tj. konkrétním ALM nástroji). Příklad klasifikačního schématu hodnot tabulek s výčtovými prvky je znázorněn na obrázku 9.



Obrázek 9: Ilustrace klasifikačního schématu výčtových hodnot

Phase

Doména *Phase* obsahuje záznamy o fázích pro ty projekty, jejichž metodologie je využívá. Každá fáze má svůj milník, který je definovaný v doméně *Milestone*. Další sloupce odpovídající tabulky z fyzického datového modelu použitelné pro grafické rozhraní, jsou *name* pro název fáze, *startDate* a *endDate* (počátek a konec fáze) a také identifikátor, kterým se ke každé fázi připojují úkoly z domény *Work Unit*. Jedna fáze může obsahovat více úkolů. Domény *Iteration* a *Activity* jsou podobné doméně *Phase* s tím rozdílem, že pro ně nejsou definovány milníky.

Work Unit

Doména *Work Unit* obsahuje úkoly, ze kterých se projekt či segment skládá. V této doméně je největší počet údajů použitelných k zobrazení v grafickém rozhraní, které je cílem této práce. Z odpovídající tabulky datového modelu lze vybrat sloupec *startDate* (počátek zpracovávání úkolu), *estimatedTime* (odhadovaná doba dokončení úkolu) a *spentTime* (doba skutečně strávená na úkolu). Dále *Work Unit* obsahuje v odpovídající tabulce z fyzického modelu řadu identifikátorů pro mapování dalších domén. Konkrétně budou potřeba identifikátory fáze (*phaseId*), iterace (*iterationId*) nebo aktivity (*activityId*). Dalšími vhodnými identifikátory hodnot z jiných tabulek jsou *priorityId*, *severityId*, *statusId*, *wuTypeId*, *resolutionId* a *assigneeId*.

Důležitým atributem je také identifikátor autora úkolu. Doménu *Work Unit* rozšiřuje doména *Work Item*, kterou dále rozšiřuje doména *Authored Entity*. V té je uložen identifikátor autora (*authorId*).

Configuration

Doména *Configuration* obsahuje záznamy o jednotlivých konfiguracích (tzn. stavech po změně artefaktu nebo úkolu v ALM nástroji – např. commitech ve VCS nástroji, úpravě

dat úkolu, doplnění wiki stránky atd.) v projektu. Jednotlivé konfigurace jsou připojeny k projektu relací 1:N. Projekt tedy může obsahovat více konfigurací. Každá konfigurace může mít několik větví (*Branch*) a několik štítků (*tags*).

Artifact

Artifact je soubor z VCS nebo jiný artefakt (email, wiki stránka), připojený ke konkrétnímu úkolu či konfiguraci. Jedna konfigurace může obsahovat více artefaktů. Zároveň jeden artefakt může být připojen k více úkolům.

Person

V doméně *Person* jsou informace o osobách, které na projektech pracují. Osoby jsou k projektu přímo vázány identifikátorem v tabulce z fyzického modelu ve sloupci *projectId*. Pro grafické rozhraní budou vhodné sloupce z fyzického modelu *id* a *name* pro identifikátor a jméno osoby. Každá osoba může v projektu zastávat několik rolí. Tyto role jsou uloženy v doméně *Role*. Role mají obdobné klasifikační schéma jako výčtové hodnoty u *Work Unit* (priorita, severita atd.).

5.1.2 Souhrn

Z analýzy datového skladu nástroje SPADe vyplynulo, že součástí grafického rozhraní musí být několik datových tříd. Hlavní datovou třídou bude třída reprezentující projekt, která bude obsahovat základní údaje z domény *Project*. Dále bude obsahovat seznamy fází, iterací, aktivit, úkolů, konfigurací, artefaktů a osob, týkající se konkrétního projektu. Artefakty ovšem nejsou přímo napojeny na doménu *Project*. V tomto případě bude třeba při výběru dat z databáze vybírat artefakty přes domény *Work Item* a *Work Unit*. Pro položky těchto seznamů je nutné vytvořit další datové třídy. Pro seznamy fází, iterací a aktivit bude stačit jedna třída pro reprezentaci segmentů, neboť instance těchto tříd budou obsahovat stejné položky pouze s rozdílným plněním. Třída pro reprezentaci úkolu bude obsahovat základní údaje jako id nebo název, a také identifikátor priority, severity, statusu, typu, resoluce či osoby, které má úkol přiřazen. Tyto možnosti úkolu budou mít své datové třídy zděděné ze stejného rodiče, protože budou mít stejné atributy (id, název, třídu a supertřídu) pouze s různým plněním. Pro třídy reprezentující priority, severity, statusy, typy, resoluce a osoby bude stačit jeden rodič s abstraktní metodou pro načítání dat, ve které si zděděná třída zvolí konkrétní typ plnění položek. Mimo třídy reprezentující projekt budou seznam svých úkolů obsahovat i třídy segmentů pro

zobrazení grafu s počtem úkolů podle fází, iterací či aktivit. Seznam úkolů bude obsahovat i třída pro reprezentaci artefaktu.

5.2 Grafické rozhraní

5.2.1 Požadavky grafického rozhraní

GUI navrhovaného nástroje musí obsahovat část pro statistické údaje a část pro grafy. Pro data, zobrazená v těchto dvou částech, musí existovat možnost filtrování. Aplikace tedy musí obsahovat část s ovládacími prvky pro tvorbu filtrů. Kritéria filtrování musí jít kombinovat. Ovládání musí být intuitivní a jednoduché. Data musí jít filtrovat podle úkolů, priorit, severit, statusů, typů, resolucí, osob, fáze, iterací, aktivit, konfigurací a artefaktů. Tyto filtry musí mít možnost filtrování podle konkrétní hodnoty (například vybrat konkrétní úkol). Tam, kde je to možné, musí být možnost filtrování podle *class* a *superclass*. U některých filtrů je nutné mít filtr na hodnoty podle konkrétního data vytvoření. V takovém případě musí být možnost načtení pouze těch hodnot, jejichž datum vytvoření se rovná (popř. nerovná, je větší, menší nebo je v rozmezí) datu zadanému ve filtru.

Úseky pro statistické údaje a pro grafy jsou stěžejním prvkem celé aplikace. Tyto části musejí být přehledné a uživatelsky přívětivé. Statistiky a grafy musí dostatečně zvýrazňovat podstatná data a musí mít jasnou vypovídající hodnotu. Všechny grafy, u nichž to má smysl, musí mít možnost vynechat ze zobrazení nulové sloupce kvůli přehlednosti. Seznam potřebných statistik a grafů byl dodán vedoucím projektu nástroje SPADe. Ve statistické části je nutné uvést údaje:

- název projektu,
- časové rozpětí,
- počet:
    - fází,
    - iterací,
    - aktivit,
    - úkolů,
    - konfigurací,
    - tagů,
    - větví,

    - artefaktů,
    - osob;
- minimum, průměr a maximum pro:
    - počet aktivit na fázi a na iteraci,
    - počet úkolů na fázi, iteraci, aktivitu a člověka,
    - odhadovaný a skutečně strávený čas na úkolu,
    - počet konfigurací na člověka,
    - počet artefaktů na člověka a konfiguraci.

V části grafů musí být grafy:

- Ganttův diagram pro fáze, iterace a aktivity,
- počet úkolů podle priorit, severit, statusů, typů a resolucí (tyto grafy musí být podle hodnoty z ALM nástroje, podle *class* a *superclass*),
- počet úkolů podle fáze, iterace, autora a osoby, která má úkol přiřazen,
- počet úkolů rostoucí v čase podle data vytvoření a graf zobrazující poměr mezi odhadovanou a skutečně strávenou dobou nad úkolem,
- počet konfigurací a artefaktů podle autora, histogramy podle data vytvoření a počet rostoucí v čase podle data vytvoření.

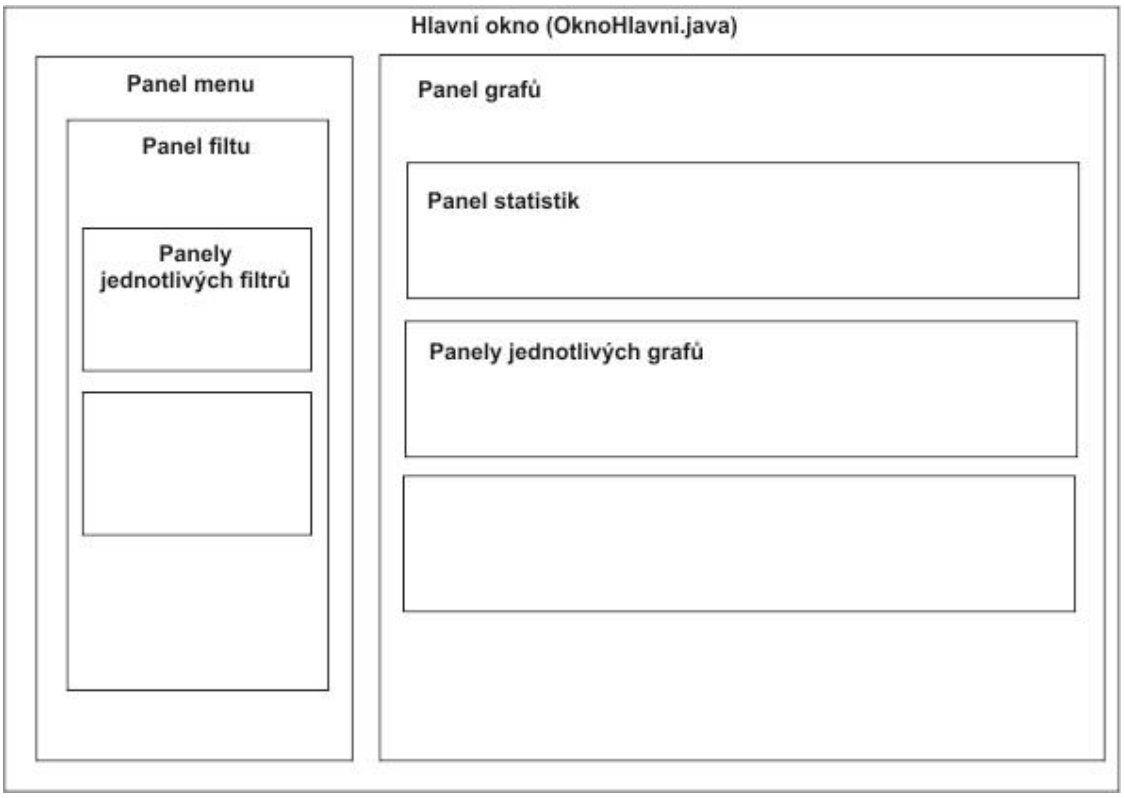
5.2.2 Popis grafického rozhraní

Navrhovaný nástroj je desktopová aplikace programovaná v jazyce Java. Hlavní okno aplikace je rozděleno na levý a pravý panel.

Levá část okna obsahuje možnosti nastavení výběru dat z databáze nástroje SPADe. Nechybí zde možnost pro výběr konkrétního projektu a filtry omezující zobrazení grafů v pravé části okna. Vzhledem k většímu počtu možných filtrů je nutné mít možnost odebrat či přidat panel s konkrétním filtrem.

Pravá část okna obsahuje panel statistických údajů a jednotlivé grafy v závislosti na vybraném projektu a nastavení filtrů v levém panelu.

Rozložení hlavního okna zobrazuje obrázek číslo 10.



Obrázek 10: Rozvržení hlavního okna

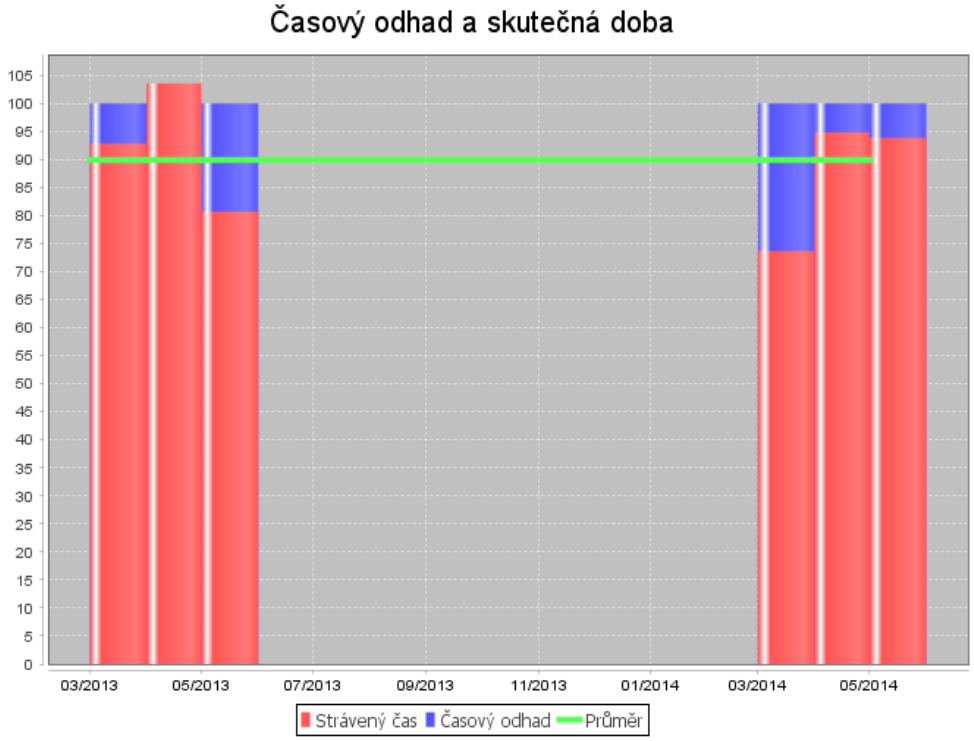
Statistický panel se zobrazuje jako první a obsahuje všechny požadované údaje a statistiky vypsané v předchozí kapitole. Panel statistik je rozdělen na část s údaji o projektu, část s počty daných prvků a část pro minima, průměry a maxima hodnot.

Po statistickém panelu následují panely s jednotlivými grafy. První panel obsahuje tři Ganttovy diagramy, zobrazující průběh projektu podle fází, iterací a aktivit.

Následuje panel s grafy pro jednotlivé úkoly projektu. Nejprve jsou zobrazeny sloupcové grafy s počty úkolů podle priorit, severit, statusů a resolucí. Pro tyto položky se zobrazují grafy podle hodnoty, tříd a supertříd. Barevné rozlišení sloupců ve všech těchto grafech určuje supertřída. Typ úkolu supertřídu nemá (má pouze jednoúrovňovou klasifikaci). Zobrazí se tedy pouze dva grafy, kde barevné zobrazení sloupců určuje třída. Dalším zobrazeným grafem je graf, který zobrazuje počet úkolů přiřazených osobám. Barvu sloupců tohoto grafu určuje velikost sloupce. Maximální počet úkolů, které jsou někomu přiřazeny, se rozdělí na třetiny. Sloupce, jejichž hodnota spadá do první třetiny, se zobrazí zeleně. Sloupce spadající do druhé třetiny se vytisknou žlutě a sloupce třetí třetiny červeně. Je tak na první pohled zřejmé, kdo má přiřazeno hodně úkolů a komu lze nějaké úkoly přidat.

Další sloupcové grafy tohoto panelu zobrazují počet úkolů podle fází a iterací. Barevné rozlišení sloupců je stejné jako u grafu s počtem úkolů přiřazených osobám.

Následuje graf zobrazující dobu strávenou nad úkoly. Každý úkol má uložen odhad doby a skutečně strávený čas. Z těchto dvou veličin je vypočteno procentuální vyjádření skutečně strávené doby v porovnání s odhadem. Toto procentuální vyjádření zobrazuje červený sloupec. Modrý sloupec určuje časový odhad, který je brán jako 100%. Viditelnost části modrého sloupce značí, že byly úkoly vyřešeny rychleji ve srovnání s počátečními odhady. Naopak viditelnost pouze červeného sloupce značí, že byly úkoly vyřešeny později oproti počátečním odhadům. Součástí tohoto grafu je i zelená linie zobrazující průměr procentuálního vyjádření. Příklad grafu zobrazujícího strávenou dobu nad úkoly ukazuje obrázek 11.



Obrázek 11: Časový odhad a skutečná doba

Předposledním grafem této skupiny je rostoucí spojnicový graf počtu úkolů v projektu. Poslední graf zobrazuje počet úkolů rozdělených podle autorů. Barvy sloupců se určují stejným způsobem, jako tomu bylo u grafu s počtem úkolů přiřazených osobám.

Další dva panely grafů jsou si podobné. Jeden panel obsahuje grafy podle konfigurací a druhý podle artefaktů. Vždy se nejprve zobrazí histogram počtu daných prvků podle data vytvoření, a poté spojnicový graf zobrazující rostoucí kumulativní počet konfigurací či artefaktů v čase. Poslední graf zobrazuje počet podle autorů konfigurací či artefaktů.

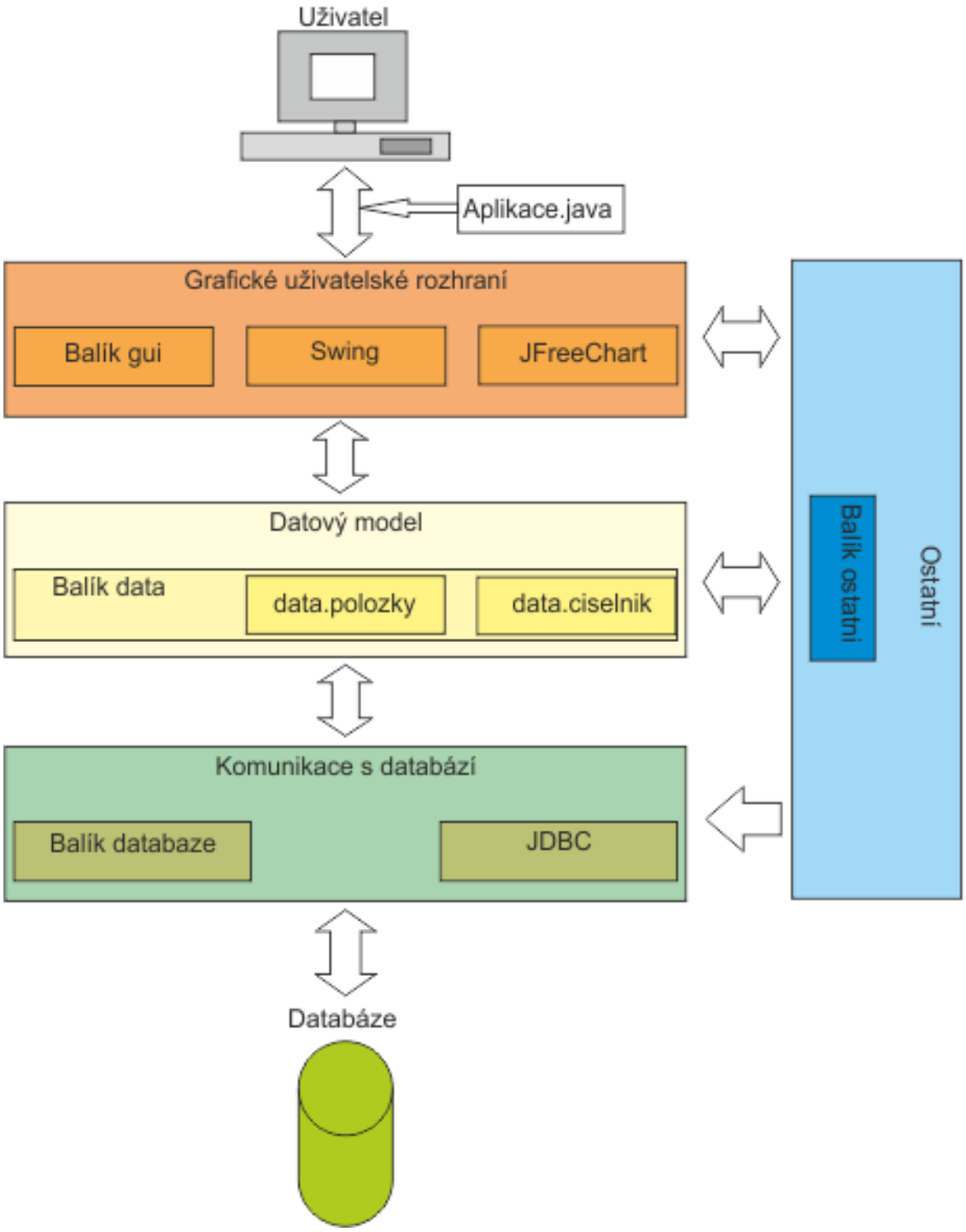
6 Implementace

Ačkoli je navrhovaná aplikace rozšířením nástroje SPADe, nepoužívá nic z implementace tohoto nástroje. Celý kód (mimo externích knihoven) je vlastní dílo autora této práce.

Navrhovaný nástroj bude desktopová aplikace. Porovnáním knihoven pro vykreslení grafů vyšla jako nejlepší možnost knihovna JFreeChart (viz kapitola 4) [13] [14] [15]. Od této volby se odvíjely i další použité technologie jako je programovací jazyk Java nebo knihovna Swing [24] [25] [26], pro kterou je JFreeChart primárně určena. Data pro grafické rozhraní jsou uložena v MySQL databázi. K databázi se aplikace připojí pomocí rozhraní JDBC (Java Database Connectivity) [27].

Třídy navrhované aplikace jsou rozděleny do balíků `data`, `data.ciselnik`, `data.polozky`, `databaze`, `gui` a `ostatni`.

Celkový pohled na architekturu aplikace ukazuje obrázek 12.



Obrázek 12: Architektura aplikace

6.1 Datový model

Kapitola popisuje balíky implementace, které obsahují třídy reprezentující datový model aplikace, plněný daty z databáze SPADe.

6.1.1 Balík data

Balík `data` obsahuje datové třídy `Projekt`, `Ukol`, `Segment`, `Artefakt`, `Konfigurace` a `Osoba`. V těchto datových třídách jsou uloženy nejdůležitější informace z datového skladu nástroje SPADe pro grafické rozhraní.

Všechny třídy v tomto balíku obsahují metody `nactiData()` a `nactiData(<podmínky>)`, ve kterých se načtou jednotlivé položky seznamů. Jedinou výjimkou je třída `Ukol`, která neobsahuje žádné seznamy a není tedy co načítat. Tyto dvě metody jsou vždy na konci třídy, z důvodu snadnějšího nalezení. Metodou `nactiData()` se načtou všechna data z databáze týkající se dané třídy. Metoda `nactiData(<podmínky>)` načítá pouze ta data, která odpovídají podmínkám. Jako podmínky jsou zadány seznamy povolených identifikátorů (`ArrayList<Integer>`), které nastavuje uživatel v panelu filtrů v levé části okna grafického rozhraní.

Například: Uživatel v panelu filtrů nastaví zobrazení pouze úkolů s určitým datem počátku. Panel filtrů zjistí, jaké úkoly tomuto nastavení filtrů odpovídají a spustí znovunačtení projektu s podmínkou povolených identifikátorů úkolů `nactiData(seznamIdUkolu, null, null, null, null, null, null, null, null, null, null)`. Pokud je seznam povolených identifikátorů null, jako jsou v tomto případě seznamy `seznamIdPriorit`, `seznamIdSeverit` atd., podmínka se pro tyto seznamy neuplatňuje.

Projekt

Je to datová třída sloužící k uložení informací o jednotlivých projektech. Třída obsahuje atributy `id` a `nazev` pro identifikátor a název projektu. Dalšími atributy jsou data pro počátek a ukončení projektu (`datumPocatku` a `datumKonce`). Dále třída obsahuje seznamy pro úkoly, fáze, iterace, aktivity, konfigurace, artefakty a osoby příslušející danému projektu. Všechny atributy jsou soukromé (`private`), ale ke každému existuje ve třídě příslušná metoda pro vrácení hodnoty. Každý seznam má ve třídě metodu `getPocetX` (například `getPocetUkolu()`), která vrací počet položek v daném seznamu. Tyto metody mají svůj význam zejména při zobrazení statistického panelu.

Díky těmto metodám je zápis čitelnější a kratší, neboť se nemusí nejprve z projektu získávat požadovaný seznam a následně zjišťovat počet položek. Na statistickém panelu bylo také nutné zobrazit počet větví a štítků, proto jsou zde i metody `getPocetVetvi()` a `getPocetTagu()`, které projdou všechny konfigurace projektu a sečtou počet větví a tagů.

Další metody, související se zobrazením statistického panelu v panelu grafů, jsou metody pro zjištění minimálních, průměrných a maximálních stavů. Pro zjištění těchto stavů je vymezeno osm soukromých metod:

- `int getMinMaxSegment(int typVypoctu, ArrayList<Segment> segmenty),`
- `double getPrumerSegment(ArrayList<Segment> segmenty),`
- `double getPredpokladanyCas(int typVypoctu),`
- `double getStravenyCas(int typVypoctu),`
- `int getMinMaxOsoba(int typVypoctu, int typSeznamu),`
- `double getPrumerOsoba(int typSeznamu),`
- `int getMinMaxKonfigurace(int typVypoctu),`
- `double getPrumerKonfigurace().`

Parametr `typVypoctu` vždy určuje, zda se jedná o zjištění minima, průměru nebo maxima. Parametr `typSeznamu` u osoby určuje, zda se hledá v osobě počet úkolů, konfigurací nebo artefaktů. Tyto soukromé metody jsou následně volány z veřejných (`public`) metod, kde se nastavením parametrů zjišťuje konkrétní hodnota. Například veřejná metoda `getMinFaze()` zavolá soukromou metodu `getMinMaxSegment(Konstanty.MINIMUM, faze)` pro vrácení minimálního počtu úkolů ve fázi.

Na konci třídy jsou metody `nactiData()` a `nactiData(<podmínky>)`. V metodě `nactiData()` se pomocí DAO (Data Access Object) tříd načítají data z databáze a ukládají se do jednotlivých seznamů. Při každém načítání dalšího seznamu se mění hodnota určující aktuální podíl již načtených dat k jejich celkovému objemu (progres; viz podkapitola OknoProgresNacitani v kapitole 6.3.1). Podle této hodnoty se mění zobrazení okna s podílem načtených dat, které běží vždy v jiném vlákně než samotné načítání dat z databáze. Metoda `nactiData(<podmínky>)` pracuje podobně, pouze předává

DAO třídám podmínky pro výběr dat a spouští navíc načítání dalších seznamů, kterých se podmínky také týkají. Podmínky metody `nactiData` jsou:

- `seznamIdUkolu`,
- `seznamIdPriorit`,
- `seznamIdSeverit`,
- `seznamIdResoluci`,
- `seznamIdStatusu`,
- `seznamIdTypu`,
- `seznamIdOsob`,
- `seznamIdFazi`,
- `seznamIdIteraci`,
- `seznamIdAktivit`,
- `seznamIdKonfiguraci`,
- `seznamIdArtefaktu`.

Všechny metody mají návratovou hodnotu `ArrayList <Integer>`.

Ukol

Datová třída `Ukol` je oproti třídě `Projekt` jednoduchá, ovšem neméně důležitá. Seznamy úkolů totiž obsahují téměř všechny datové třídy z balíku `data`.

Třída je zděděná z třídy `PolozkaPocatek`, z níž dědí atributy `id`, `nazev`, `datumVytvoreni` a `datumPocatku`. K tomu přidává atributy `predpokladanyCas` a `stravenyCas` pro odhad a skutečnou dobu potřebnou pro dokončení úkolu. Dalšími atributy jsou identifikátory položek číselníků příslušející danému úkolu:

- `priorityID` – identifikátor priority úkolu,
- `severityID` – identifikátor severity úkolu,
- `statusID` – identifikátor statusu úkolu,
- `typID` – identifikátor typu úkolu,
- `resoluceID` – identifikátor resoluce úkolu,
- `prirazenID` – identifikátor osoby, která má úkol přiřazen,
- `autorID` – identifikátor autora úkolu.

Všechny atributy jsou soukromé, ale mají své metody pro vrácení jejich hodnot.

Segment

Třída `Segment` slouží k uložení informací o fázích, iteracích nebo aktivitách. Je zděděna z třídy `PolozkaPocatek`, z níž přebírá atributy `id`, `nazev`, `datumVytvoreni` a `datumPocatku`. K těmto atributům přidává atributy `typSegmentu`, `datumKonce` a `ukoly`. Atribut `typSegmentu` je typu `Integer` a určuje, zda se jedná o fázi, iteraci nebo aktivitu (konkrétní hodnoty jsou uloženy ve třídě `Konstanty` – viz kapitola 6.4.2). Atribut `datumKonce` typu `LocalDate` určuje datum ukončení segmentu a `ArrayList<Ukol> ukoly` ukládá seznam úkolů příslušejících danému segmentu. Atributy jsou soukromé, ale třída obsahuje `get-` a `set-` metodu pro každý z nich.

Artefakt

Artefakt je soubor z VCS nebo jiný artefakt (email, wiki stránka), připojený ke konkrétnímu úkolu či konfiguraci. Třída `Artefakt` je zděděna z třídy `PolozkaVytvoreni`, z níž dědí atributy `id`, `nazev` a `datumVytvoreni`. Navíc přidává atributy `typ` (textový řetězec určující typ souboru), `velikost` (určující velikost souboru) a `ukoly` (seznam úkolů, ke kterým je artefakt přiřazen).

Konfigurace

Třída ukládající informace o jednotlivých konfiguracích (stavech po změně artefaktu nebo úkolu v ALM nástroji, např. commitech ve VCS nástroji, úpravě dat úkolu, doplnění wiki stránky atd.) projektu. Je zděděna ze třídy `PolozkaVytvoreni` a k atributům `id`, `nazev` a `datumVytvoreni` přidává atributy `artefakty` (seznam artefaktů dané konfigurace), `vetve` (seznam branchů dané konfigurace) a `tagy` (seznam tagů dané konfigurace).

Osoby

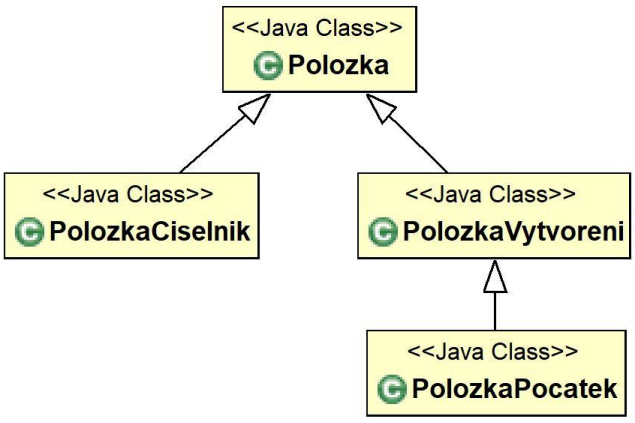
Datová třída obsahující údaje o osobách podílejících se na projektu. Třída je zděděna ze třídy `Polozka`, ze které dědí atributy `id` a `nazev`. K těmto atributům přidává seznamy úkolu přiřazených osobě (`ArrayList<Ukol> ukoly`), konfigurací provedených danou osobou (`ArrayList<Konfigurace> konfigurace`) a artefaktů vložených osobou (`ArrayList<Artefakt> artefakty`).

Ke všem seznamům existují `get-` metody pro vrácení celého seznamu, `getPocet-` metody pro vrácení počtu položek v seznamu a `add-` metody pro vložení jedné položky do seznamu. Navíc je zde metoda `getPocet(int typSeznamu)`, která vrací počet položek podle zadaného typu seznamu v parametru. Typy seznamů jsou uloženy ve třídě `Konstanty` (viz kapitola 6.4.2). Na konci třídy jsou opět metody `nactiData()` a `nactiData(<podmínky>)`.

6.1.2 Balík data.polozky

Balík `data.polozky` obsahuje datové třídy pro uložení nejzákladnějších údajů datového modelu. Tyto třídy se využívají ve všech ostatních datových třídách.

Hlavní třídou je zde třída `Polozka`. Hierarchii balíku `data.polozky` ukazuje UML (Unified Modeling Language) diagram na obrázku 13:



Obrázek 13: UML diagram balíku data.polozky

Polozka

Obsahuje atributy `id` a `nazev` pro identifikátor a název položky. Dále třída obsahuje metody pro vrácení těchto atributů („getry“) a metodu `toString()`, která vrací v současné době stejnou hodnotu jako `getNazev()`. Metoda `toString()` je zde pro možné předefinování některým z potomků, pokud to bude v budoucnu potřeba.

Tato třída se používá pro jednoduché položky obsahující pouze identifikátor a název jako jsou branche (větve) a tagy (štítky) ve třídě konfigurací.

PolozkaCiselnik

K atributům `id` a `nazev`, zděděné ze třídy `Polozka`, přidává navíc atributy `idTrida`, `trida` a `supertrida`. Tyto atributy se nastavují v konstruktoru třídy. Dále třída přidává metody pro vrácení („getry“) a nastavení („setry“) těchto atributů.

Tato třída se využívá pro položky číselníků obsahující identifikátor, název, třídu a supertřídu. Konkrétně to jsou třídy z balíku `data.ciselnik` (`Ciselnik`, `Priority`, `Severity`, `Status`, `Osoby`, `Resoluce` a `Typ`).

PolozkaVytvoreni

K atributům třídy `Polozka` přidává navíc atribut `datumVytvoreni` a příslušný getr a setr tohoto atributu.

Využívá se pro položky, které obsahují datum vytvoření, ale neobsahují datum počátku. Konkrétně tuto třídu dědí třídy `Artefakt` a `Konfigurace` z balíku `data`, které si dále přidávají další vlastní atributy.

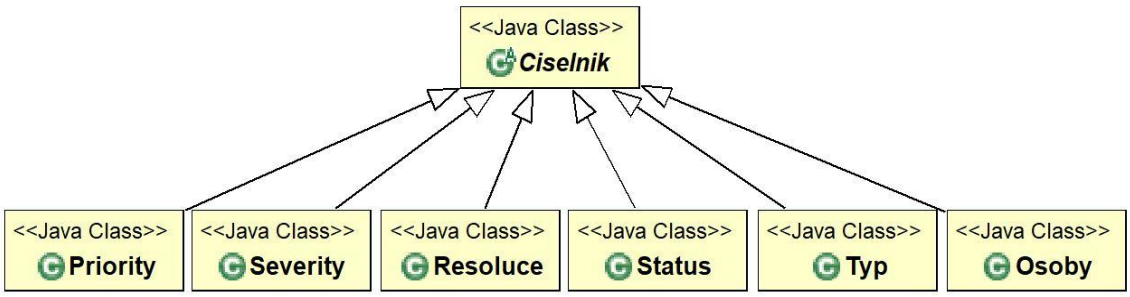
PolozkaPocatek

Třída zděděná ze třídy `PolozkaVytvoreni`. K atributům `id`, `nazev` a `datumVytvoreni` přidává navíc atribut `datumPocatku` a jeho getr a setr.

Třída se využívá pro položky obsahující identifikátor, název, datum vytvoření a datum počátku. Konkrétně tuto třídu dědí třídy `Segment` a `Ukol` z balíku `data`.

6.1.3 Balík data.ciselnik

Základem tohoto balíku je abstraktní třída `Ciselnik`, ze které ostatní třídy dědí. Dalšími třídami v tomto balíku jsou třídy `Priority`, `Severity`, `Resoluce`, `Status`, `Typ` a `Osoby`. Hierarchii balíku `data.ciselnik` zobrazuje UML diagram na obrázku 14.



Obrázek 14: UML diagram balíku data.ciselnik

Ciselnik

Je to abstraktní třída sloužící pro naplnění číselníků. Obsahuje seznam položek daného číselníku (třída `PolozkaCiselnik`) a identifikátor typu číselníku podle statické hodnoty uložené ve třídě `Konstanty` (viz kapitola 6.4.2; např. `Konstanty.Priority` = 0). Důležitou metodou v této třídě je abstraktní metoda

`nactiPolozky()`, kterou musí jednotlivé číselníky implementovat. V této metodě musí třídy spustit načtení příslušných dat. Dalšími zajímavými metodami jsou `getIdTridy(String nazev, boolean podleTridy)` a `getSuperTrida(String nazev, boolean podleTridy)`. Tyto metody podle zadaného názvu položky vrací identifikátor třídy nebo název supertřídy. Podle těchto hodnot se následně rozhoduje o barvě sloupce v příslušném grafu.

Priority, Severity, Resoluce, Status, Typ a Osoby

Všechny tyto třídy jsou zděděné ze třídy `Ciselnik`. Obsahují konstruktor, ve kterém se spouští konstruktor předka, se správným typem a identifikátorem vybraného projektu. Dále tyto třídy implementují metodu `nactiPolozky(int idProjekt)`, ve které pomocí třídy `CiselnikyDAO` (viz podkapitola CiselnikyDAO kapitoly 6.2.1), implementující rozhraní `ICiselnikyDAO`, volají konkrétní metody pro načtení položek do seznamu.

6.2 Komunikace s databází

Kapitola popisuje balík s třídami a rozhraními, zajišťující správný výběr dat z databáze nástroje SPADe.

6.2.1 Balík databaze

Balík `databaze` obsahuje DAO (Data Access Object) třídy a jejich rozhraní, zajišťující výběr dat z databáze pomocí připojení realizovaného technologií JDBC. V balíku jsou třídy `ArtefaktDAO`, `CiselnikyDAO`, `KonfiguraceDAO`, `ProjektDAO` a `UkolDAO`. K těmto třídám jsou zde i rozhraní se stejným názvem doplněným o prefix „`I-`“ na začátku (`IArtefaktDAO`, `ICiselnikyDAO`, `IKonfiguraceDAO`, `IProjektDAO` a `IUkolDAO`).

V konstruktoru každé třídy se přiřadí připojení k databázi, které se navazuje po přihlášení uživatele. Pro samotný výběr dat se používá třída `PreparedStatement` z balíku `java.sql`. Ve všech rozhraních jsou metody vracející seznam položek daného výběru. Většina těchto metod je navíc přetížená. DAO třídy musí tedy implementovat i metody, které mají navíc přidaný parametr se seznamem možných identifikátorů položek pro výstup. Tento seznam identifikátorů vychází z panelu filtrů, ve kterém uživatel určuje, jaká konkrétní data požaduje. Například ve třídě `ProjektDAO` je metoda `ArrayList<Segment> getFaze(int idProjekt)`, která vrací seznam všech

fází v projektu se zadaným identifikátorem. Ve stejné třídě je i metoda `ArrayList<Segment> getFaze(int idProjekt, ArrayList<Integer> seznamIdFazi)`, která vrací seznam všech fází v projektu, jejichž id ovšem musí být v seznamu fází v parametru, neboť tyto fáze uživatel vybral. První metoda se volá při načítání projektu, druhá při spuštění filtru.

ProjektDAO

Třída implementuje rozhraní `IProjektDAO`.

V tomto rozhraní jsou metody:

- `ArrayList<Projekt> getProjekt()` – vrací seznam projektů v databázi,
- `ArrayList<Segment> getFaze(int idProjekt)` – podle projektu vrací seznam fází,
- `ArrayList<Segment> getIterace(int idProjekt)` – podle projektu vrací seznam iterací,
- `ArrayList<Segment> getAktivity(int idProjekt)` – podle projektu vrací seznam aktivit,
- `ArrayList<Segment> getOsoby(int idProjekt)` – podle projektu vrací seznam osob.

K těmto metodám jsou přetížené metody:

- `getFaze(int idProjekt, ArrayList<Integer> seznamIdFazi)`,
- `getIterace(int idProjekt, ArrayList<Integer> seznamIdIteraci)`,
- `getAktivity(int idProjekt, ArrayList<Integer> seznamIdAktivit)`,
- `getOsoby(int idProjekt, ArrayList<Integer> seznamIdOsob)`.

V `ProjektDAO` je navíc soukromá metoda `getSegmentData(PreparedStatement stmt)`, které ostatní metody předávají vytvořený `PreparedStatement`. Metoda ho spustí a naplní `ArrayList`, který následně vrátí volající metodě.

UkolDAO

Tato třída implementuje rozhraní `IUkolDAO`. Úkol (v doménovém modelu SPADe *Work Unit*) se prolíná celým projektem a je tedy součástí více datových tříd. Proto je v tomto rozhraní více metod k implementaci. Všechny metody vrací seznam úkolů (`ArrayList<Ukol>`):

- `getUkolProjekt(int idProjekt)`,
- `getUkolFaze(int idFaze)`,
- `getUkolIterace(int idIterace)`,
- `getUkolAktivity(int idAktivity)`,
- `getUkolOsoba(int idOsoby)`,
- `getUkolArtefakt(int idArtefakt)`.

Všechny tyto metody jsou přetížené. Úkoly ovšem lze filtrovat podle více kritérií. Proto i všechny metody obsahují více seznamů identifikátorů, kterým musí úkoly odpovídat. Konkrétně se jedná o seznamy identifikátorů úkolů, priorit, severit, resolucí, statusů, typů a osob.

Samotná třída `UkolDAO` navíc implementuje soukromou metodu `getUkol(<podmínky>)`, která poskládá podmínky a spustí předpřipravený dotaz do databáze. Poté vrátí seznam úkolů volající metodě.

KonfiguraceDAO

Tato třída implementuje rozhraní `IKonfiguraceDAO`. Toto rozhraní obsahuje metody:

- `ArrayList<Konfigurace> getKonfiguraceProjekt(int idProjekt)`,
- `ArrayList<Konfigurace> getKonfiguraceOsoba(int idOsoba)`,
- `ArrayList<Polozka> getVetveKonfigurace(int idKonfigurace)`,
- `ArrayList<Polozka> getTagyKonfigurace(int idKonfigurace)`.

Metody `getKonfiguraceProjekt(int idProjekt)` a `getKonfiguraceOsoba(int idOsoba)` jsou navíc přetížené o seznam povolených identifikátorů konfigurací.

Třída `KonfiguraceDAO` navíc implementuje soukromé metody `getKonfigurace` a `getPolozka`, kterým ostatní metody předávají parametry a předpřipravené dotazy do databáze. Tyto metody sestaví podmínky a spustí předpřipravený dotaz. Následně vrátí volající metodě seznam položek či konfigurací.

ArtefaktDAO

Třída `ArtefaktDAO` implementuje rozhraní `IArtefaktDAO`. Všechny metody tohoto rozhraní vrací seznam artefaktů (`ArrayList<Artefakt>`):

- `getArtefaktyProjekt(int idProjekt)`,
- `getArtefaktyKonfigurace(int idKonfigurace)`,
- `getArtefaktyOsoba(int idOsoby)`.

Všechny tyto metody jsou přetížené o seznam povolených identifikátorů artefaktů.

Třída `ArtefaktDAO` navíc implementuje metodu `getArtefakty(String sql, int id, ArrayList<Integer> seznamIdArtefaktu)`, která poskládá podmínky a spustí předpřipravený dotaz do databáze. Následně vrací volajícímu seznam artefaktů.

CiselnikyDAO

Třída `CiselnikyDAO` implementuje rozhraní `ICiselnikyDAO`. Všechny metody vrací seznam položek číselníku (`ArrayList<PolozkaCiselnik>`):

- `getPriority(int idProjekt)`,
- `getSeverity(int idProjekt)`,
- `getStatus(int idProjekt)`,
- `getTyp(int idProjekt)`,
- `getResoluce(int idProjekt)`,
- `getOsoby(int idProjekt)`.

Tyto metody již nejsou dále přetěžovány.

Třída `CiselnikyDAO` navíc implementuje metodu `getData(String sql, int idProjekt)`, kterou ostatní metody volají. Metoda nastaví parametry dotazu do databáze, spustí ho a vrátí seznam položek číselníku volajícímu.

6.3 GUI

V tomto balíku jsou třídy grafického rozhraní. Balík obsahuje čtyři základní typy tříd:

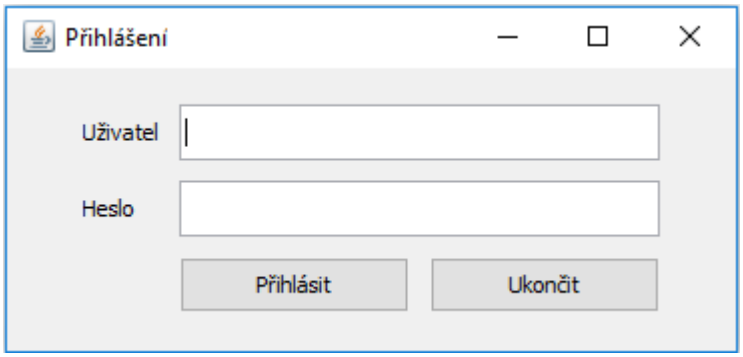
- třídy pro vykreslení oken:
  - `OknoHlavni`,
  - `OknoPrihlasovani`,
  - `OknoProgresNacitani`;
- třídy pro vykreslení panelů filtrů na hlavním okně:
  - `PanelFiltr`,
  - `PanelFiltrCiselnik`,
  - `PanelFiltrKonfigurace`,
  - `PanelFiltrPolozkaPocatek`,
  - `PanelFiltrPolozkaVytvoreni`;
- třídy pro vykreslení panelů grafů na hlavním okně:
  - `PanelGrafu`,
  - `PanelGrafuArtefakt`,
  - `PanelGrafuKonfigurace`,
  - `PanelGrafuRodic`,
  - `PanelGrafuSegment`,
  - `PanelGrafuUkol`;
- ostatní třídy:
  - `ComboBoxDynamicka`,
  - `RendererSloupcovyGraf`.

6.3.1 Třídy pro vykreslení oken

Do této skupiny tříd se řadí třídy `OknoHlavni`, `OknoPrihlasovani` a `OknoProgresNacitani`, které vykreslují okna aplikace.

OknoPrihlasovani

Instance této třídy se vytvoří okamžitě při spuštění aplikace. Zobrazuje přihlašovací okno do aplikace pro zadání uživatele a hesla, který má přístup do databáze. Zobrazené okno ukazuje obrázek 15.

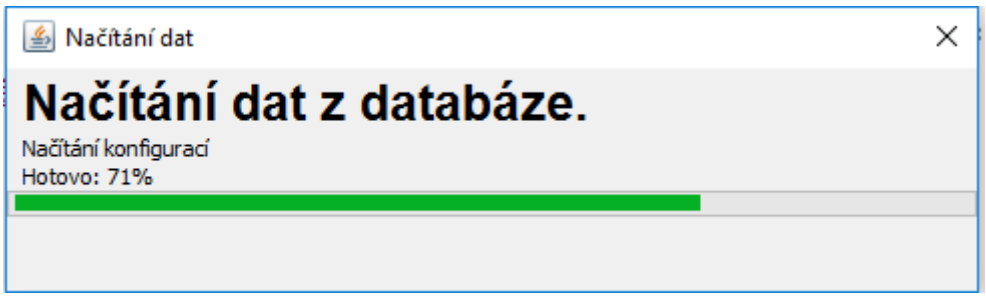


Obrázek 15: Okno pro přihlášení

Třída obsahuje pouze konstruktor a dvě soukromé metody. V konstruktoru se nastaví okno a ostatní komponenty. Poté se spustí soukromá metoda `nastavAkce()`, ve které se vytvoří a přiřadí akce po stisknutí tlačítek „*Přihlásit*“ a „*Ukončit*“. V akci pro tlačítko „*Přihlásit*“ se nejprve zkontroluje, zda je zadán uživatel a heslo. Pokud ano, spustí se druhá soukromá metoda v této třídě, a to metoda `prihlasit(String login, String heslo)`. V této metodě se nejprve vytvoří spojení s databází pomocí rozhraní JDBC. Pokud se spojení vytvoří úspěšně, vytvoří se instance hlavního okna. Z důvodu velkého objemu dat v databázi ale načítání chvíli trvá. Je tedy třeba uživateli sdělit, co se děje. Proto se po navázání spojení s databází vytvoří dvě vlákna. Jedno vlákno spouští hlavní okno a druhé zobrazuje okno s progresem načítání dat. Ve druhém vláknu se okno s progresem neustále aktualizuje do doby, než načítání dat skončí.

OknoProgresNacitani

Toto okno se spouští při načítání dat z databáze. Zobrazuje aktuální progres a popis, jaká data se momentálně načítají. Zobrazené okno ukazuje obrázek 16.

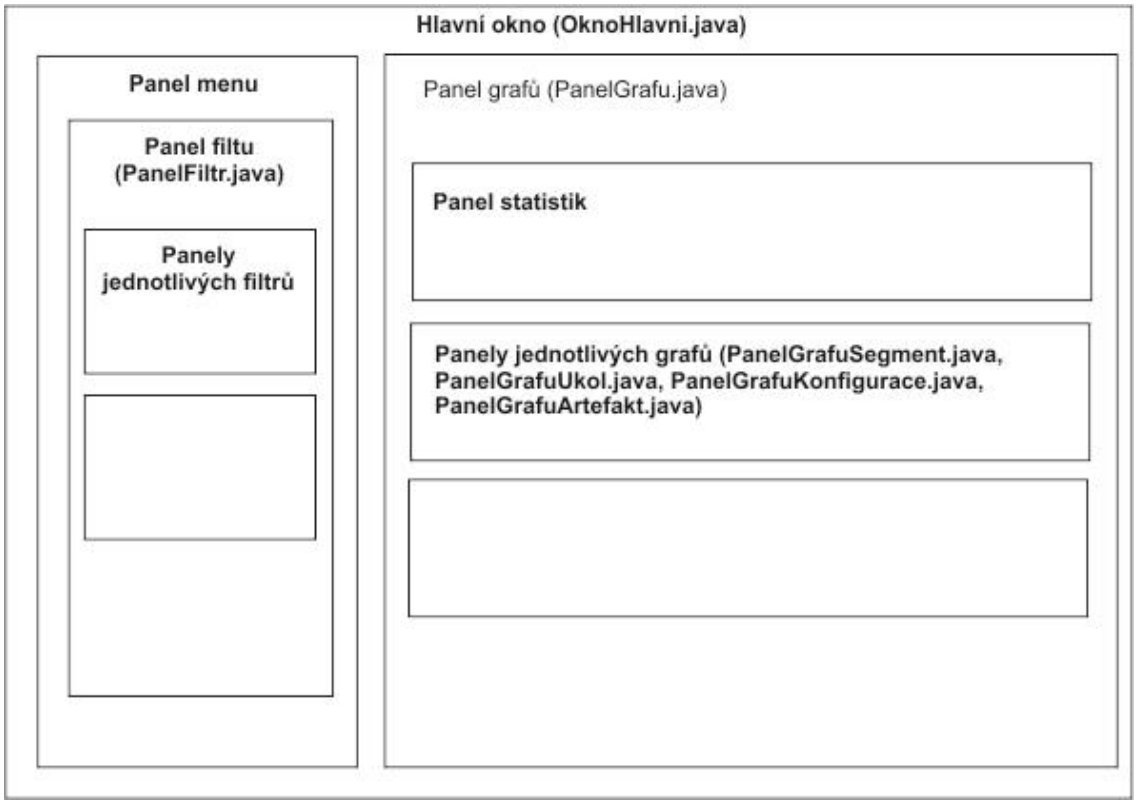


Obrázek 16: Okno s progresem načítání

Tato třída obsahuje dva konstruktory. Konstruktor `OknoProgresNacitani()` se spouští po přihlášení do aplikace, neboť v tuto chvíli nemá okno žádného předka. Konstruktor `OknoProgresNacitani(Component rodic)` se spouští při změně projektu v panelu menu hlavního okna. Dále třída obsahuje soukromou metodu `nastavZobrazeni()`, která se spouští v obou těchto konstruktorech. Tato metoda nastavuje okno a jeho komponenty. Poslední metodou v této třídě je veřejná metoda `nastavProgres()`. Po spuštění této metody se nastaví velikost zobrazeného progresu podle hodnoty `CITAC_PROGRESU` ve třídě `Konstanty` (viz kapitola 6.4.2). Podle velikosti čítače se nastaví i ostatní popisy.

OknoHlavni

Tato třída zobrazuje hlavní okno celé aplikace. V tomto okně se řídí zobrazení jednotlivých panelů s grafy či filtry. Rozvržení panelů na hlavním okně ukazuje obrázek 17 a má zjevnou závislost na návrh rozložení hlavního okna, viz obrázek 10.



Obrázek 17: Rozvržení panelů na hlavním okně

Konstruktoru této třídy předává přihlašovací okno připojení k databázi. Na prvním místě konstruktoru se toto připojení nastaví do proměnné `PRIPOJENI` ve třídě `Konstanty`

(viz kapitola 6.4.2). Dále se spustí soukromé metody `nactiProjekty()` a `nastavZobrazeni()`.

V metodě `nactiProjekty()` se načtou z databáze projekty pro uložení do comboboxu (rozbalovacího seznamu) se seznamem projektů v panelu menu. Data projektů pro zobrazení grafů (např. seznamy úkolů, fází, konfigurací atd.) si načítá každý projekt sám až ve chvíli, kdy je vybrán pro zobrazení. Při spuštění hlavního okna se tedy načtou data pouze pro první projekt v seznamu.

V metodě `nastavZobrazeni()` se nejprve spustí metoda `nastavOkno()`, která nastaví zobrazení okna. Dále se v metodě `nastavZobrazeni()` vytvoří a nastaví panel grafů a panel menu. V této metodě se spouští také soukromá metoda `nastavAkce()`, která k jednotlivým komponentám doplní jejich akce.

V metodě `nastavAkce()` se nejprve nastavuje akce při změně projektu v panelu menu. V této akci se vytvářejí dvě vlákna. V jednom vlákně se spouští okno s progresem načítání a ve druhém vlákně se přiřadí panelu grafů nový projekt, čímž se opětovně spouští načítání dat z databáze.

Dále se nastavuje akce pro přidání filtru do panelu filtrů. Nejprve se zjistí, jaký filtr se má přidat podle hodnoty comboboxu. Soukromou metodou `jeZadanyFiltr(String nazevFiltru, Component[] ulozeneKomponenty)` se zjistí, zda již filtr v panelu není. Pokud ne, vloží se nový filtr do panelu.

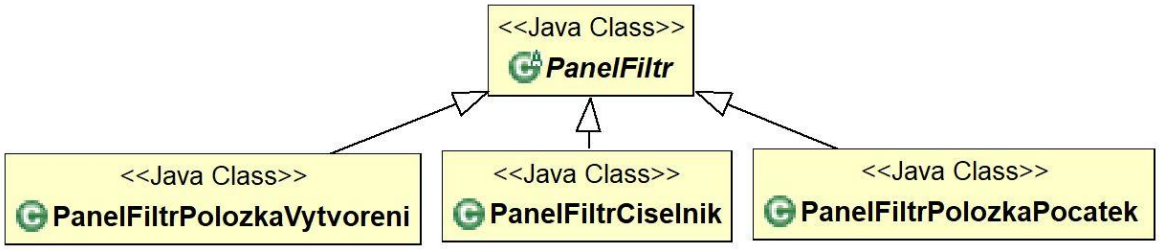
Akce pro tlačítko „*Zapni Filtr*“ spouští opět dvě vlákna. V prvním vlákně se spustí okno s progresem načítání a znepřístupní se veškerá tlačítka, dokud nedoběhne načítání dat. Ve druhém vlákně se nejprve (pomocí metody panelů filtrů `getSeznamId()`) zjistí seznamy identifikátorů z jednotlivých filtrů, které odpovídají zadaným podmínkám. Následně se tyto podmínky předají panelu grafů metodou `setPodminkyProjektu(<podmínky>)`, který poté spustí načítání dat s odpovídajícími parametry.

Dále se zde nastavuje akce pro změnu velikosti panelu grafů při změně velikosti okna a akce pro ukončení spojení s databází při zavření okna.

6.3.2 Třídy pro vykreslení panelů filtrů na hlavním okně

Do této skupiny tříd se řadí třídy `PanelFiltr`, `PanelFiltrCiselnik`, `PanelFiltrPolozkaPocatek` a `PanelFiltrPolozkaVytvoreni`. Pomocí

těchto tříd se zobrazují filtry v panelu menu aplikace. Hlavní třídou tohoto balíku je třída `PanelFiltr`, kterou ostatní třídy dědí vzhledem k podobným atributům filtrování, a tudíž i jeho možnostem. Hierarchii tříd zobrazuje UML diagram na obrázku 18.



Obrázek 18: UML diagram tříd panelů filtrů

PanelFiltr

`PanelFiltr` je abstraktní třída, kterou ostatní třídy panelů filtrů dědí. V konstruktoru se předává seznam (`ArrayList`) hodnot, které filtr obsahuje. Důležitou metodou je `nastavPanel(String nazev)`, která je typu `protected`. Tato metoda podle zadaného názvu nastaví titulek panelu filtru. Poté dodá do filtru zaškrtávátko, které určuje, zda se má filtr použít, a combobox se seznamem hodnot pro výběr. Všechny třídy zděděné z `PanelFiltr` budou mít tyto dvě možnosti. Dále se zde spouští abstraktní metoda `nastavAkce()`, kterou musí ostatní třídy implementovat. Další abstraktní metodou k implementaci je třída `ArrayList<Integer> getSeznamId()`, která má vracet seznam identifikátorů odpovídajících zadaným filtrům.

PanelFiltrCiselnik

Tato třída zděděná z `PanelFiltr` se používá pro panely filtrů priorit, severit, statusů, typů, resolucí a osob. V konstruktoru spouští funkci `nastavPanel(String nazev)`, ve které přidává ke komponentám z `PanelFiltr` navíc možnosti pro výběr tříd a supertříd.

V metodě `nastavAkce()` se nastavují akce k zaškrtávátku a seznamům. Po změně zaškrtnutí se hodnoty komponent nulují a po vybrání seznamu se zpřístupní seznam možných spojení podmínek.

Důležitou metodou je zde `ArrayList<Integer>getSeznamId()`, která vrací seznam identifikátorů odpovídající podmínkám. V metodě se prochází celý seznam položek a pokud položka odpovídá vybraným kritériím, vloží se její identifikátor do výstupního seznamu. Používá se zde metoda `jeVSeznamu(PolozkaCiselnik`

`polozka, int typVypoctu)`, která zjišťuje, zda aktuálně prohledávaná položka je vybrána v jednom ze seznamů.

PanelFiltrPolozkaPocatek

Tato třída se používá pro zobrazení panelů filtrů položek, které obsahují datum počátku, tedy úkolů a segmentů (fází, iterací a aktivit). K původním komponentám z `PanelFiltr` přidává v metodě `nastavPanel(String nazev)` datum od a do, podle kterých lze omezit úkoly či segmenty začínající v daném časovém úseku.

V metodě `getSeznamId()` se podle operátoru data rozhoduje, jakou metodou se zjišťuje, zda konfigurace odpovídá zadaným podmínkám filtru. Na výběr jsou metody:

- `vlozitDoSeznamuMezi(PolozkaPocatek polozka)`,
- `vlozitDoSeznamuVetsi(PolozkaPocatek polozka)`,
- `vlozitDoSeznamuVetsiRovno(PolozkaPocatek polozka)`,
- `vlozitDoSeznamuRovno(PolozkaPocatek polozka)`,
- `vlozitDoSeznamuNerovno(PolozkaPocatek polozka)`,
- `vlozitDoSeznamuMensioRovno(PolozkaPocatek polozka)`,
- `vlozitDoSeznamuMensi(PolozkaPocatek polozka)`.

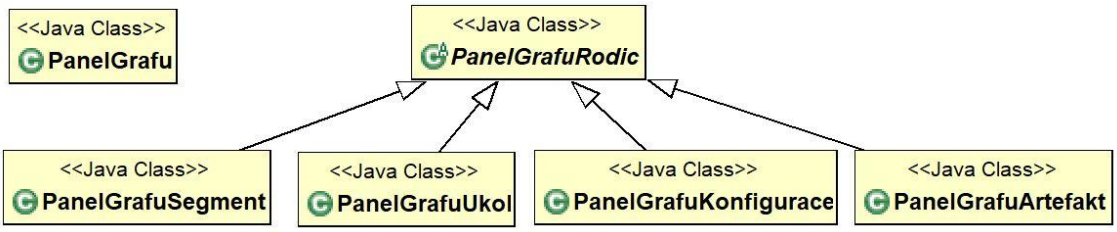
Všechny tyto metody nejprve zjišťují, zda datum počátku položky odpovídá zadaným datům. Následně kontrolují pomocí metody `jeVSeznamu(PolozkaPocatek polozka)`, zda je položka vybrána v seznamu položek.

PanelFiltrPolozkaVytvoreni

Tato třída se používá pro zobrazení panelů filtrů položek, které obsahují datum vytvoření, tedy konfigurací a artefaktů. Třída je podobná třídě `PanelFiltrPolozkaPocatku`, rozdíl je pouze v kontrole dat. Tato třída kontroluje datum vytvoření místo data počátku.

6.3.3 Třídy pro vykreslení panelů grafů na hlavním okně

Do této skupiny se řadí třídy `PanelGrafu`, `PanelGrafuRodic`, `PanelGrafuSegment`, `PanelGrafuUkol`, `PanelGrafuKonfigurace` a `PanelGrafuArtefakt`. Tyto třídy slouží k zobrazení panelů grafů v pravé části okna aplikace. Třída `PanelGrafu` představuje hlavní panel, na který se ostatní panely skládají. Další důležitou třídou je `PanelGrafuRodic` ze které jsou ostatní třídy zděděné. Hierarchii tříd zobrazuje UML diagram na obrázku 19.



Obrázek 19: UML diagram panelů grafů

PanelGrafu

Třída `PanelGrafu` je zděděná ze třídy `JPanel`. V konstruktoru vyvolá načtení dat aktuálně vybraného projektu. Následně spustí metodu `nastavZobrazeni()`, ve které na sebe poskládá všechny panely grafů. Panel se statistikami se tvoří přímo v této třídě pomocí soukromé metody `getPopisProjektu()`.

Další důležitou metodou je `setPodminkyProjektu(<podmínky>)`, která vyvolá načtení dat projektu podle zadaných podmínek filtrů.

PanelGrafuRodic

Je to abstraktní třída panelů grafů zděděná ze třídy `JPanel`. Tuto třídu ostatní třídy dědí. V konstruktoru nastavuje aktuální projekt. Obsahuje abstraktní metodu `vlozGrafy()`, kterou musí ostatní panely grafů implementovat. V metodě `nastavZobrazeni()` se nastavuje stejná velikost všech grafů v panelu. Další metodou je `nastavGraf(JFreeChart graf, int typGrafu)`, ve které se podle typu grafu nastaví jeho zobrazení. Metody `getAutorGraf(int typVypoctu, boolean bezNul)` a `getSpojnicovyGrafPocet(ArrayList seznam, String nazevGrafu)` vytvoří dva grafy (počet položek autora a počet položek v průběhu času), které se zobrazují téměř na všech panelech. Dále třída obsahuje metodu `vlozCheckBoxBezNulAutor(ChartPanel chartPanel, int typVypoctu)`, která vloží zaškrtávací tlačítko do kontextového menu grafu autorů. Toto zaškrtávací tlačítko „*Bez nulových hodnot*“ zajistí zobrazení grafu pouze se sloupci, jejichž hodnoty jsou nenulové.

PanelGrafuSegment

Třída implementuje metodu `vlozGrafy()`, ve které vkládá na panel tři Ganttovy diagramy (fáze, iterace a aktivity). Obsahuje soukromou metodu `getGanttGraf(int typGrafu)`, ve které tyto grafy vytváří.

PanelGrafuUkol

Třída implementuje metodu `vlozGrafy()`, ve které vkládá na panel 20 různých grafů. Metody pro vytvoření těchto grafů jsou:

- `getCiselnikGraf(Ciselnik seznam, int typGrafu, boolean bezNul)` – vytvoří grafy pro položky číselníků (za normální hodnoty, třídy a supertřídy),
- `getSegmentGraf(ArrayList<Segment> seznam, int typGrafu, boolean bezNul)` – vytvoří grafy pro fáze, iterace a aktivity (graf za aktivity se v tomto panelu nevkládá, ale do třídy tato možnost byla vložena pro případné pozdější využití),
- `getHistogramCasy()` – vytvoří graf s přehledem časového odhadu a skutečně strávené doby nutné k dokončení úkolu.

Metody `vlozCheckBoxBeNulCiselnik(ChartPanel chartPanel, Ciselnik seznam, int typGrafu)` a `vlozCheckBoxBezNulSegment(ChartPanel chartPanel, ArrayList<Segment> seznam, int typGrafu)` vloží k těmto sloupcovým grafům zaškrtávací tlačítko pro zobrazení pouze nenulových sloupců.

PanelGrafuKonfigurace

Tato třída se využívá k zobrazení panelu s grafy týkající se konfigurací. Implementuje metodu `vlozGrafy()`. Další metodou je `getHistogramKonfigurace()`, která vytvoří histogram konfigurací podle data vytvoření.

PanelGrafuArtefakt

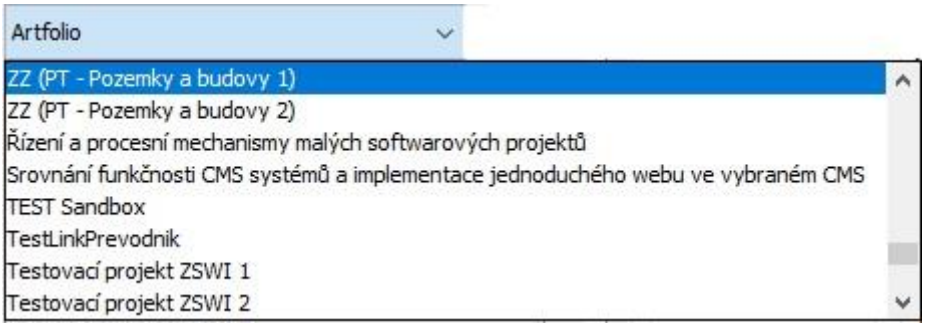
Třída se využívá k zobrazení panelu s grafy artefaktů. Implementuje metodu `vlozGrafy()` a obsahuje metodu `getHistogramArtefakty()`, která vytvoří histogram artefaktů podle data vytvoření.

6.3.4 Ostatní třídy

Do této skupiny patří třídy `ComboBoxDynamicky` a `RendererSloupcovyGraf`.

ComboBoxDynamicky

Třída zděděná ze třídy `JComboBox`. Využívá se pro seznam projektů v panelu menu. Původní třídu `JComboBox` upravuje tak, aby při rozbalení seznamu byly zobrazeny celé názvy projektů. Rozbalený combobox je zobrazen na obrázku 20.



Obrázek 20: ComboBoxDynamicky

RendererSloupcovyGraf

Třída zděděná z třídy `BarRenderer`. Tato třída udává barevné zobrazení sloupcových grafů.

6.4 Ostatní třídy

Mezi ostatní třídy se řadí třída `Aplikace` a třídy v balíku `ostatni`.

6.4.1 Třída Aplikace

Třída `Aplikace` je spouštěcí třída s metodou `main`. V té se nejprve načte soubor s popiskami aplikace a poté se spustí přihlašovací okno.

6.4.2 Balík ostatni

V tomto balíku je pouze třída `Konstanty`. Ve třídě `Konstanty` jsou uloženy hodnoty využívající se v celé aplikaci.

6.5 Ostatní aspekty implementace

Aplikace je v současné době v češtině, ale je připravena na pozdější lokalizaci do jiných jazyků. Při spuštění aplikace se načítá soubor *czech.properties* v němž jsou uloženy všechny texty.

Samotná knihovna JFreeChart poskytuje mnoho funkcí, které zlepšují práci s vytvořeným nástrojem. Například zvětšení konkrétní oblasti v grafu je velmi užitečným pomocníkem. Další zajímavou možností je rozšiřitelné kontextové menu. K možnostem, které zde již

jsou (tisk grafu, export do obrázku ve formátu PNG, atd.), lze přidat další. Díky tomu nebylo třeba dodávat v aplikaci další ovládací prvky na panel grafu a tím ho znepřehlednit. Například checkbox (zaškrtávací políčko) pro zobrazení grafů „*Bez nulových hodnot*“ byl dodán přímo do kontextového menu určitých grafů. Pozdější práce s aplikací ukáže, zda není třeba rozšířit menu o další ovládací prvky.

Samotná velikost panelu s grafy se automaticky mění při změně velikosti okna. Tím lze rolovat mezi grafy i na menších monitorech.

7 Testování

Testování probíhalo metodou bílé i černé skříňky. Metodu bílé skříňky využíval vývojář během psaní kódu. Při tomto testování byly využity jednotkové (JUnit) testy. K tomuto účelu bylo nutné dočasně změnit modifikátory přístupu některých atributů a metod na `public`. Jednotkových testů je celkem 29 a pokrytí kódu je 71,3 %.

Pro metodu černé skříňky bylo využito testovací středisko firmy Alfa Software, s.r.o. Aby testeři nezjistili přihlašovací údaje do databáze, bylo pro účely testování v kódu zakomentováno přihlašování.

Šest testerů bylo rozděleno do třech skupin po dvojicích, z nichž první skupina prováděla testování pomocí předpřipravených scénářů. Druhá skupina prováděla zátěžové testování a monkey testy (testování založené na náhodném klikání do aplikace). Tito testeři byli seznámeni s obsluhou programu. Třetí skupina nebyla s obsluhou seznámena a zkoumala intuitivnost a uživatelskou přívětivost aplikace.

Testovaná aplikace nevykazovala v průběhu testování větší problémy. Většina problémů, které testeři nalezli, se týkaly neúplnosti dat v databázi (například resoluce jsou pouze u jednoho projektu, artefakty nejsou připojeny k úkolům atd.). Chyby, které skutečně souvisely s aplikací, byly opraveny a následně testeři provedli regresní testy (testy zkoumající, zda nová verze programu nezhoršila původní funkčnost). Program je intuitivní a lze ho ovládat bez předešlého zaškolování. Aplikace neměla problémy ani při rychlém sledu přepínání projektů či zadávání filtrů. V závislosti na rychlosti připojení bylo potřeba různých časů pro načtení dat z databáze.

Akceptační testy byly prováděny vedoucím bakalářské práce v průběhu psaní kódu v rámci konzultačních schůzek.

8 Budoucí rozšíření aplikace

U aplikace se počítá s budoucím rozšířením. Velký prostor pro rozšiřování aplikace je u samotných grafů. Lze přidat další panely s grafy pro větší pokrytí datového modelu, komplexnější grafy a statistiky, agregované grafy pro data více projektů zároveň atd. Hlavním kandidátem na další graf určitě bude rozdělení osob pracujících na projektu podle rolí. S tím souvisí i rozšíření datové třídy pro reprezentaci osob. Při přidávání dalších grafů bude možné využít i další typy grafů popsaných v kapitole 3. Například ve výsečovém grafu by se mohlo zobrazit procentuální zastoupení úkolů podle jednotlivých tříd a supertříd priorit, severit, statusů atd. S přidáváním grafů lze rozšiřovat i jejich konfigurovatelnost přidáním dalších možností v kontextovém menu grafů nebo přidáním dalších filtrů. Pro tyto účely je kód programu podrobně okomentován.

Dalším budoucím rozšířením aplikace může být cizojazyčná lokalizace, ke které v podstatě stačí vytvořit properties soubor a změnit název načítaného souboru při spuštění okna. V budoucnu by mohl být v hlavním okně combobox se seznamem podporovaných jazyků, ve kterém by si mohl uživatel lokalizaci sám měnit.

V případě možného budoucího převodu na webovou aplikaci doporučuji využít nástroj Google Charts, který vyšel ze srovnání velmi dobře. Z důvodu responzivnosti by bylo vhodné využít framework Bootstrap [28].

9 Závěr

Cílem práce bylo vytvoření uživatelského rozhraní prezentujícího obsah datového skladu nástroje SPADe. Tím měla rozšířit funkčnost nástroje SPADe a do budoucna zlepšit a usnadnit identifikaci a prezentaci procesních chyb (anti-patternů) ve vedení softwarových projektů (hlavní účel nástroje SPADe).

Ke splnění daných cílů bylo nutné seznámit se s účelem nástroje SPADe a s daty, která uchovává (kapitola 2), a dále prozkoumat možnosti zobrazení informací o obsahu jeho datového skladu. Výstupem této části práce je souhrnný popis základních typů grafů, u nichž je předpoklad využití v uživatelském rozhraní, ať již v současné době, nebo při pozdějším rozvoji aplikace (kapitola 3), a srovnání knihoven použitelných pro vykreslení těchto grafů v desktopové i webové aplikaci (kapitola 4). Jako nejvhodnější byla vybrána knihovna JFreeChart pro desktopovou aplikaci. Pro případnou pozdější implementaci webového rozhraní byl jako nejvhodnější identifikován nástroj Google Charts.

Druhým výsledkem práce je analýza datového skladu nástroje SPADe, výběr vhodných dat k zobrazení a návrh rozložení grafického uživatelského rozhraní ve vyvíjené aplikaci (kapitola 5).

Posledním a hlavním výstupem této bakalářské práce je vlastní aplikace včetně popisu její implementace (kapitola 6) a testování (kapitola 7). Důkladnější popis implementace pak představují dokumentační komentáře v kódu a uživatelská příručka v příloze B. Rovněž byly identifikovány některé možnosti dalších rozšíření aplikace pro její budoucí vývoj (kapitola 8).

Ač práce spadá do širšího projektu SPADe, do kterého budou v budoucnu její výstupy integrovány, na jeho současnou implementaci nijak nenavazuje. Výjimkou jsou jen zvolený programovací jazyk Java a databáze SPADe, ze které aplikace čerpá data. Práce je tak (s výjimkou použitých knihoven třetích stran a citovaných zdrojů) zcela původním dílem autora.

Při implementaci uživatelského rozhraní si autor vyzkoušel nové techniky se kterými se dosud nesetkal, jako je implementace grafů pomocí různých knihoven či připojení k MySQL databázi pomocí JDBC atd. Tyto techniky budou pro autora přínosem v dalších letech studia i v praxi.

Bakalářská práce splňuje zadání ve všech bodech a je připravena k obhajobě.

Seznam zkratek

ALM	–	Application Lifecycle Management
API	–	Application Programming Interface
BSD	–	Berkeley Software Distribution
CCM	–	Configuration and Change Management
CSS	–	Cascading Style Sheets
DAO	–	Data Access Object
DWH	–	Data Warehouse
EPS	–	Encapsulated PostScript
ETL	–	Extract – Transform – Load
FAV	–	Fakulta aplikovaných věd
GIF	–	Graphics Interchange Format
GNU GPL	–	GNU's Not Unix General Public License
Gral	–	GRAphing Library
GUI	–	Graphical User Interface
JCCKit	–	Java Chart Construction Kit
JDBC	–	Java Database Connectivity
JPEG	–	Joint Photographic Experts Group
KIV	–	Katedra informatiky a výpočetní techniky
LGPL	–	Library General Public License
MySQL	–	My Structured Query Language
PDF	–	Portable Document Format
PHP	–	Hypertext Preprocessor, Personal Home Page
PNG	–	Portable Network Graphics
SPADe	–	Software Process Anti-patterns Detector
SVG	–	Scalable Vector Graphics
SVN	–	Subversion
UML	–	Unified Modeling Language
URL	–	Uniform Resource Locator
VCS	–	Version Control Systém
ZČU	–	Západočeská univerzita v Plzni

Seznam obrázků

Literatura

[1] P. Pícha, *Datový model nástroje SPADe a jeho mapování na projektová data z ALM nástrojů*, referát, Západočeská univerzita v Plzni, 2016.

[2] P. Pícha a P. Brada, *ALM Tool Data Usage in Software Process Metamodeling*, 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, 2016.

[3] P. Pícha, P. Brada, R. Ramsauer a W. Mauerer, *Towards Architect's Activity Detection Through a Common Model for Project Pattern Analysis*, 1st International Workshop on the Social and Organizational Dimensions of Software, IEEE, 2017.

[4] A. L. Dennis, *The Data Warehouse: From the Past to the Present* [online]. 2017. Dostupné na: http://www.dataversity.net/data-warehouse-past-present/.

[5] T. Kratěna, *Portlet pro interaktivní vizualizaci dynamických dat*, diplomová práce, Brno: Masarykova univerzita, Fakulta informatiky, 2013, vedoucí diplomové práce RNDr. Radek Ošlejšek, Ph.D.

[6] S. F. Rahman, *16 JavaScript Libraries for Creating Beautiful Charts* [online]. 2015. Dostupné na: https://www.sitepoint.com/15-best-javascript-charting-libraries/.

[7] FromDev, *10 Excellent Free Open Source Java Chart Library for Developers* [online]. 2012. Dostupné na: http://www.fromdev.com/2012/09/Free-Open-Source-Java-Charting-Library.html.

[8] K. Radhakrishnan, *Top 15 Java Charting and Reporting Tools For Developers* [online]. 2016. Dostupné na: http://toppersworld.com/top-15-java-charting-and-reporting-tools-for-developers/.

[9] *What is the best open-source java charting library? (other than jfreechart)* [online], fórum. 2016. Dostupné na: https://stackoverflow.com/questions/265777/what-is-the-best-open-source-java-charting-library-other-than-jfreechart.

[10] *Free java library for visualization* [online], fórum. 2011. Dostupné na: https://stackoverflow.com/questions/8284975/free-java-library-for-visualization-plots-charts-graphs.

[11] E. Seifert, *GRAL Java Graphing* [online]. 2016. Dostupné na: http://trac.erichseifert.de/gral/.

[12] JCCKit, *Chart Construction Kit for the Java platform*, [online]. 2004. Dostupné na: http://jcckit.sourceforge.net/.

[13] M. Čimbora, *Knižnica JFreeChart pre vytváranie grafov*, bakalářská práce, Brno: Masarykova univerzita, Fakulta informatiky, 2012, vedoucí bakalářské práce Mgr. Marek Grác.

[14] JFree, *JFreeChart*, [online]. 2014. Dostupné na: http://www.jfree.org/jfreechart/.

[15] Tutorials Point, *JFreeChart Tutorial* [online]. 2017. Dostupné na: https://www.tutorialspoint.com/jfreechart/.

[16] Krysalis Community Project, *jCharts* [online]. 2004. Dostupné na: http://jcharts.sourceforge.net/.

[17] Approximatrix, *Openchart2* [online]. 2017. Dostupné na: https://approximatrix.com/products/openchart2/.

[18] Google, *Google Charts* [online]. 2017. Dostupné na: https://developers.google.com/chart/.

[19] B. Jahoda, *Vytváření grafů v JavaScriptu* [online]. 2014. Dostupné na: http://jecas.cz/grafy.

[20] *Chart.js*, [online]. Dostupné na: http://www.chartjs.org/.

[21] *Chartist.js Simple responsive charts*, [online]. Dostupné na: https://gionkunz.github.io/chartist-js/.

[22] J. D. Pogolotti, *pChart* [online]. 2017. Dostupné na: http://www.pchart.net/.

[23] Asial Corporation, *JpGraph* [online]. Dostupné na: http://jpgraph.net/.

[24] P. Conrod a L. Tylee, *Learn Java Gui Application: A JFC Swing Tutorial*, Kidwa Software LLC, 2013, ISBN 978-1-937161-55-2.

[25] D. Čápka, *Okenní aplikace v Java Swing* [online]. 2017. Dostupné na: https://www.itnetwork.cz/java/swing.

[26] Tutorials Point, *Swing Tutorial* [online]. 2017. Dostupné na: http://www.tutorialspoint.com/swing/.

[27] G. Reese, *Database programming with JDBC and Java*, O'Reilly, 2000, ISBN 1-56592-616-1.

[28] J. Spurlock, *Bootstrap*, O'Reilly, 2013, ISBN 978-1-4493-4391-0.

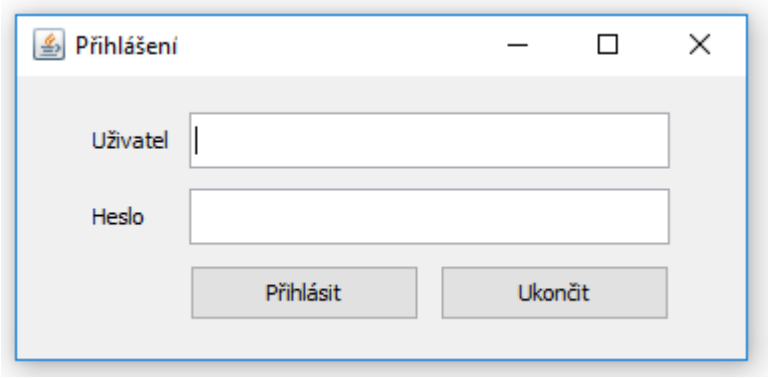
Přílohy

A. Obsah CD

- **Dokumentace**
  - ***Bakalářská práce.docx*** – tato bakalářská práce ve formátu DOCX
  - ***Bakalářská práce.pdf*** – tato bakalářská práce ve formátu PDF
  - ***Javadoc*** – dokumentace aplikace
- **Fyzický datový model SPADe**
  - ***SPADe.mwb*** – fyzický datový model SPADe ve formátu MWB
  - ***SPADe.png*** – fyzický datový model SPADe ve formátu PNG
- **Projekt**
  - ***.settings***
  - ***lib*** – externí knihovny aplikace
  - ***bin*** – přeložené class soubory
  - ***src*** – zdrojové kódy
- **Spustitelný program**
  - ***GUISpade.jar*** – spustitelná aplikace
  - ***czech.properties*** – soubor textů pro aplikaci v češtině
- **Testování**
  - ***JUnit testy*** – zdrojové kódy použitých JUnit testů
  - ***Scénáře*** – scénáře pro testování ve formátu PDF
- **CTI_ME.txt** – obsahuje tento rozpis obsahu CD nosiče

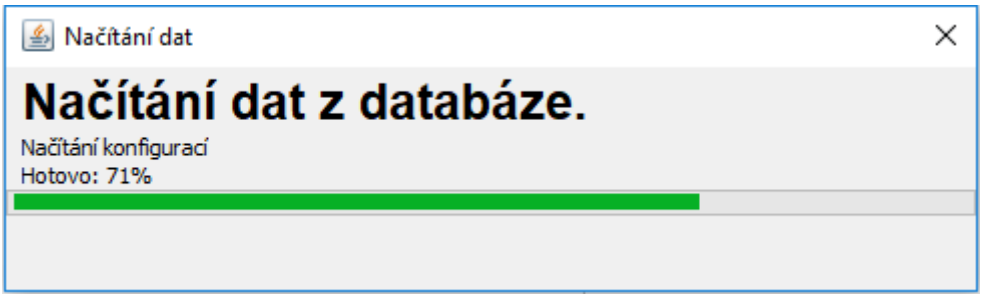
B. Uživatelská dokumentace

Po spuštění aplikace se zobrazí přihlašovací okno pro zadání uživatelského jména a hesla.



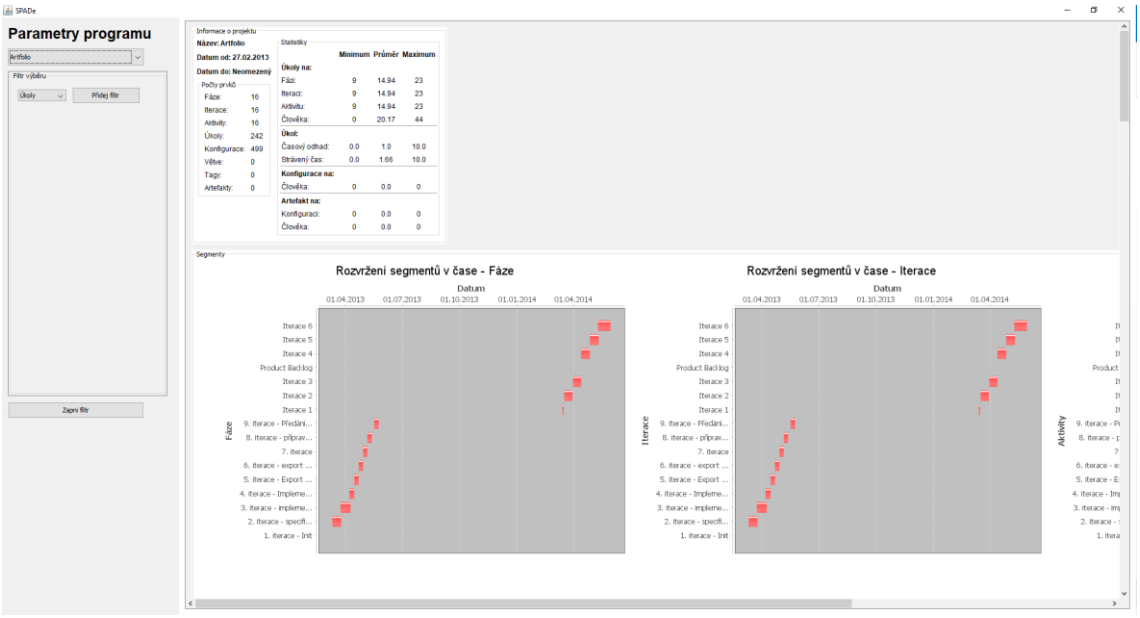
Obrázek 21: Přihlašovací okno

Tlačítkem „*Ukončit*“ se aplikace ukončí. Po zadání platného názvu uživatelského účtu a hesla se lze přihlásit tlačítkem „*Přihlásit*“. Po ověření zadaných údajů se spustí načítání dat z databáze, jehož průběh je signalizován oknem „*Načítání dat*“.



Obrázek 22: Okno načítání dat

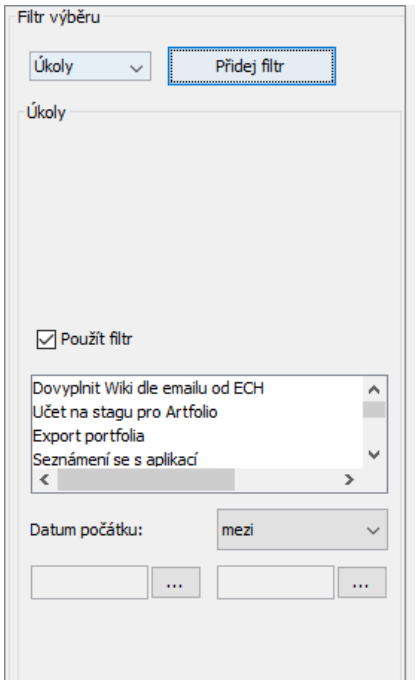
Po dokončení načítání dat se přes celou obrazovku spustí hlavní okno aplikace.



Obrázek 23: Spuštěná aplikace

Okno je rozdělené na dvě části. V levé části je panel s parametry pro nastavení zobrazení dat. Jako první parametr je seznam projektů, ze kterého lze vybrat konkrétní projekt k zobrazení. Po změně projektu v seznamu se automaticky spustí načítání dat.

Další parametr je seznam možných filtrů. Ze seznamu se vybere konkrétní filtr a kliknutím na tlačítko „*Přidej filtr*“ se doplní panel s konkrétními možnostmi k omezení dat.

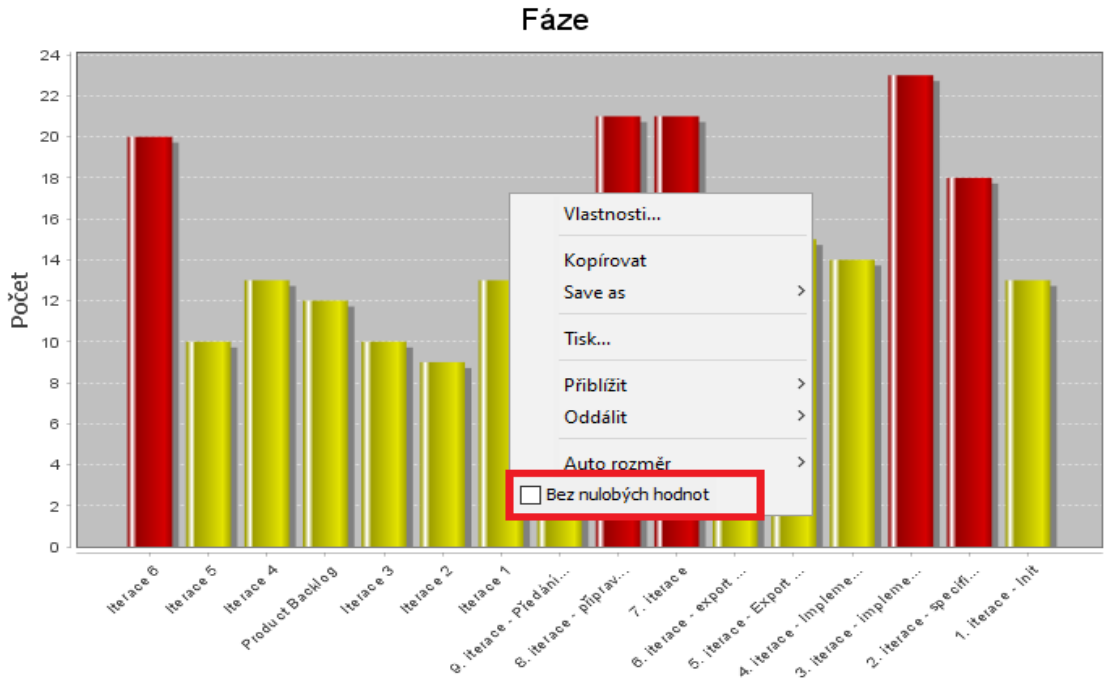


Obrázek 24: Panel filtrů

Každý panel filtrů má zaškrtávátko „*Použít filtr*“. Při změně zaškrtnutí se vynulují všechny předešlé výběry ve filtru. Panelů s filtry lze zadat více, ovšem z každého typu pouze jeden. V programu existuje několik typů panelů filtrů. Všechny panely mají seznam položek k výběru (např. seznam úkolů, priorit atd.). U položek, u kterých je to možné, jsou přidány navíc seznamy tříd a supertříd. Tyto seznamy mají navíc možnost vybrání operátoru spojení podmínek („*a*“, „*nebo*“). Ve všech seznamech lze vybírat více možností najednou. U některých panelů lze nastavit rozmezí položek podle data počátku nebo vytvoření pomocí zadání dvou dat a konkrétního operátoru („*mezi*“, „<“, „<=“, „=“, „*!*=“, „>=“, „>“).

Po stisknutí tlačítka „*Zapni filtr*“ se zaktualizují data v pravé části okna a filtry, které nemají zaškrtnuté „*Použít filtr*“, se schovají.

V pravé části jsou panely se statistikami a grafy dat odpovídající zadaným parametrům. V grafech lze přibližovat konkrétní úseky označením dané oblasti držením a tahem levého tlačítka myši. K původnímu zobrazení se lze vrátit držením a tahem levého tlačítka myši ve směru nahoru v oblasti grafu. Kliknutím pravého tlačítka myši v oblasti grafu se zobrazí kontextové menu. U sloupcových grafů je přidáno na konci kontextového menu zaškrtávátko „*Bez nulových hodnot*“. Po jeho zaškrtnutí se v grafech zobrazí pouze sloupce, které mají nějaké hodnoty.



Obrázek 25: Zaškrtávátko „Bez nulových hodnot“

Ukončení programu se provádí křížkem v pravém horním rohu.

C. Ukázky scénářů testování

TS.01 Funkčnost

Test Case BC-3: TC.01.01 Spuštění programu [Version : 1]		
Author:	admin - 06/01/2017 01:45:44	
Summary: Happy day scenario		
#:	Step actions:	Expected Results:
1	Spustí se program	Otevře se okno Přihlášení
2	Vyplní se správné přihlašovací údaje	
3	Kliknutí na tlačítko Přihlásit	Zavře se okno Přihlásit a spustí se okno s progresem načítání
4	Doběhne načítání	Okno s progresem se zavře a spustí se hlavní okno

TS.02 Okno přihlášení

Test Case BC-1: TC.02.01 Přihlašovací okno [Version : 1]		
Author:	admin - 06/01/2017 00:41:48	
Summary: Kontrola rozvržení a viditelnosti komponent přihlašovacího okna		
#:	Step actions:	Expected Results:
1	Spuštění programu	Zobrazí se okno Přihlášení
2	Okno má titulek Přihlášení	
3	Existuje pole pro zapsání loginu	
4	Pole pro zapsání loginu má titulek Uživatel	
6	Existuje pole pro zapsání hesla	
7	Pole pro zapsání hesla má titulek Heslo	
8	Existuje tlačítko Přihlásit	
9	Existuje tlačítko Ukončit	

TS.03 Okno progresu načítání

Test Case BC-2: TC.03.01 Okno progresu načítání [Version : 1]		
Author:	admin - 06/01/2017 01:07:08	
Summary: Kontrola viditelnosti komponent okna prograsu		
Preconditions: Jsou vyplněny správné údaje v přihlašovacím okně a klikne se na tlačítko přihlášení		
#:	Step actions:	Expected Results:
1	Zobrazí se okno s progresem načítání dat	
2	Okno má titulek načítání dat	
3	V okně je nadpis Načítání dat z databáze.	
4	V okně je popis, jaká data se právě načítají	
5	Popis obsahuje postupně text:	
6		Načítání úkolů
7		Načítání fází
8		Načítání iterací
9		Načítání aktivit
10		Načítání osob
11		Načítání konfigurací
12		Načítání artefaktů
13	Existuje popis "Hotovo:" s procentuálním vyjádřením načtených dat	
14	Existuje progressbar se zelenou lištou	
15	Zelená lišta zobrazuje progres v závislosti na načtených datech	
16	Po skončení načítání se okno zavře	

TS.04 Hlavní okno

TS.04.01 Rozložení komponent

Test Case BC-5: TC.04.01.01 Rozložení okna [Version : 1]		
Author:	admin - 06/01/2017 05:42:15	
Summary: Rozložení komponent v okně		
Preconditions: Je spuštěno grafické rozhraní nástroje SPADe, uživatel je přihlášen		
#:	Step actions:	Expected Results:
1	Titulek okna obsahuje text SPADe	
2	V levé části je panel s nadpisem Parametry programu	
3	Pod nadpisem je combobox se seznamem projektů	
4	Při rozbalení seznamu jsou vidět celé názvy projektů	
5	V panelu existuje panel filtrů s popisem Filtr výběru	
6	Existuje combobox se seznamem filtrů	
7	Seznam filtrů obsahuje filtry:	
8	Úkoly	
9	Priority	
10	Severity	
11	Statusy	
12	Typy	
13	Resoluce	
14	Osoby	
15	Fáze	
16	Iterace	
17	Aktivity	
18	Konfigurace	
19	Artefakty	
20	Existuje tlačítko Přidej filtr	
21	Existuje tlačítko Zapni filtr	
22	V hlavní části existuje panel statistik s nadpisem Informace o projektu	
23	Panel statistik obsahuje název, datum počátku a konce projektu	
24	Panel statistik obsahuje počty prvků:	
25	Fáze	
26	Iterace	
27	Aktivity	
28	Úkoly	
29	Konfigurace	
30	Větve	

31	Tagy	
32	Artefakty	
33	Panel statistik obsahuje minimum, maximum a průměr pro:	
34	Úkoly na fázi	
35	Úkoly na iteraci	
36	Úkoly na aktivitu	
37	Úkoly na člověka	
38	Časový odhad	
39	Strávený čas	
40	Konfigurace na člověka	
41	Artefakty na konfiguraci	
42	Artefakty na člověka	
43	Pod panelem statistik existuje panel s grafy segmentů	
44	Panel obsahuje grafy:	
45	Rozvržení segmentů v čase - Fáze	
46	Rozvržení segmentů v čase - Iterace	
47	Rozvržení segmentů v čase - Aktivity	
48	Pod panelem segmentů existuje panel s grafy úkolů	
49	Panel obsahuje grafy:	
50	Priority, Priority - Třídy a Priority - Supertřídy	
51	Severity, Severity - Třídy a Severity - Supertřídy	
52	Statusy, Statusy - Třídy a Statusy - Supertřídy	
53	Resoluce, Resoluce - Třídy a Resoluce - Supertřídy	
54	Typy a Typy - Třídy	
55	Přiřazeno	
56	Fáze	
57	Iterace	
58	Časový odhad a skutečná doba	
59	Počet úkolů	
60	Autor úkolu	
61	Pod panelem úkolů existuje panel s grafy konfigurací	
62	Panel obsahuje grafy:	
63	Počet konfigurací podle data vytvoření	
64	Počet konfigurací rostoucí v čase	
65	Autor konfigurace	
66	Pod panelem konfigurací existuje panel s grafy artefaktů	
67	Panel obsahuje grafy:	
68	Počet artefaktů podle data vytvoření	
69	Počet artefaktů rostoucí v čase	
70	Autor artefaktu	

TS.04.02 Funkčnost okna

Test Case BC-7: TC.04.02.02 Filtr úkolů [Version : 1]		
Author:	admin - 06/21/2017 00:36:07	
Summary: Zobrazení a funkčnost filtru úkolů		
Preconditions: Je spuštěná aplikace. Je vybrán projekt podle vlastního uvážení.		
#:	Step actions:	Expected Results:
1	V seznamu filtrů se vybere Úkoly	
2	Stiskne se tlačítko Přidej filtr	Zobrazí se panel Úkoly
3		Panel obsahuje zaškrtnuté zaškrtávátko Použít filtr
4		Panel obsahuje seznam úkolů k vybrání
5		Panel obsahuje komponentu pro zadání počátečního data
6		Panel obsahuje komponentu pro zadání koncového data
7		Panel obsahuje seznam operátorů datumů
8	V seznamu úkolů jsou vybrány dva úkoly	
9	Je stisknuto tlačítko Zapni filtr	Zobrazí se okno s progresem načítání
10	Doběhne okno s progresem načítání	Zobrazí se grafy a statistiky obsahující pouze dva zadané úkoly
11	Vyberou se data konce a počátku úkolů podle grafů Časový odhad a Počty úkolů	
12	Zadají se do filtru vybrané datumy a vybere se operátor "mezi"	
13	Stiskne se tlačítko Zapni filtr	Zobrazí se okno s progresem načítání
14	Doběhne okno s progresem načítání	Zobrazí se úkoly spadající do zadaného intervalu
15	Vybere se operátor ">" v panelu filtru	Schová se komponenta pro zadání data do
16	Stisk tlačítka Zapni filtr	Zaktualizují se grafy a statistiky
17	Vybere se operátor ">=" a stiskne se Zapni filtr	Zaktualizují se grafy a statistiky
18	Vybere se operátor "=" a stiskne se Zapni filtr	Zaktualizují se grafy a statistiky
19	Vybere se operátor "!=" a stiskne se Zapni filtr	Zaktualizují se grafy a statistiky
20	Vybere se operátor "<=" a stiskne se Zapni filtr	Zaktualizují se grafy a statistiky
21	Vybere se operátor "<" a stiskne se Zapni filtr	Zaktualizují se grafy a statistiky
22	Odškrtne se zaškrtávátko Použít filtr	
23	Klikne se na tlačítko Zapni filtr	Zobrazí se grafy a statistiky pro všechna data v projektu a schová se panel filtru pro úkoly

D. Fyzický datový model