Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Detekce špatných praktik projektového řízení v open-source projektech

Plzeň 2022 Petr Štěpánek

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Petr ŠTĚPÁNEK**
Osobní číslo:	**A19B0615P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační systémy**
Téma práce:	**Detekce špatných praktik projektového řízení v open-source projektech**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s problematikou špatných praktik a anti-vzorů v projetkovém řízení vývoje softwaru a projektem SPADe.
2. Analyzujte data vybraných open-source projektů a vyberte špatné praktiky vhodné pro detekci v těchto projektech.
3. Navrhněte modely vybraných praktik a případná rozšíření současné implementace SPADe pro jejich automatickou detekci.
4. Implementujte detekci špatných praktik a rozšíření z předchozího bodu, ověřte na datech vybraných projektů a výsledky zhodnoťte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Petr Pícha**
Nové technologie pro informační společnost

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

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

V Plzni dne 23. června 2022

Petr Štěpánek

Abstract

The aim of this bachelor thesis is to model and detect selected bad project management practices in data of open-source projects using the framework Software Process Anti-pattern Detector (SPADe), which includes the SPADe Web Interface (SPAWn) application. The SPAWn application is used to detect anti-patterns and bad practices using project data in the SPADe data warehouse. For the purpose of this work, SPAWn application has been extended with functional features and a set of automatic detections of bad practices. Furthermore, an experiment was performed on a set of data from selected open-source projects.

Abstrakt

Cílem této bakalářské práce je modelovat a detekovat vybrané špatné praktiky projektového řízení v datech open-source projektů s využitím frameworku Software Process Anti-pattern Detector (SPADe), jehož součástí je aplikace SPADe Web Interface (SPAWn). Aplikace SPAWn slouží k detekci anti-vzorů a špatných praktik z projektových dat v datovém skladu SPADe. Za účelem této práce byla aplikace SPAWn rozšířena o funkční vlastnosti a sadu automatických detekcí špatných praktik. Následně byl proveden experiment nad množinou dat vybraných open-source projektů.

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Petru Píchovi za odborné vedení, cenné rady a čas, který mi věnoval v průběhu celé práce.

Obsah

1 Úvod

Projekt vývoje softwaru zahrnuje množství různých aktivit pro dosažení stanoveného cíle. Kromě samotného vývoje zahrnuje například i analýzu požadavků, tvorbu specifikace, návrh řešení či testování. Do projektu je zpravidla zapojeno více účastníků, kteří mohou být různě specializovaní, a jejichž správné řízení a koordinace jsou klíčovými prvky pro úspěšné dokončení projektu. Pokud řízení projektu není zvládnuto správně, může to mít negativní dopad na průběh projektu, zejména pak na výsledný produkt, který tak nemusí být dodán v požadované kvalitě nebo v určeném čase. Často se jedná o opakující se chyby známé jako špatné praktiky či anti-vzory. Včasné upozornění na tyto chyby by mohlo významně přispět k úspěšnému dokončení projektu.

Problematiku špatných praktik a anti-vzorů v projektovém řízení vývoje softwaru řeší framework Software Process Anti-pattern Detector (SPADe) vyvíjený na Katedře informatiky a výpočetní techniky (KIV) Fakulty aplikovaných věd (FAV) Západočeské univerzity v Plzni (ZČU). SPADe získává data z Application Lifecycle Management (ALM) nástrojů a uchovává je v jednotné struktuře pro účely následné analýzy. Součástí frameworku SPADe je podpůrná aplikace SPADe Web Interface (SPAWn) provádějící detekce vybraných anti-vzorů a poskytující webové rozhraní.

Cílem této práce je detekovat špatné praktiky v projektových datech vybraných open-source projektů s využitím aplikace SPAWn. Za tímto účelem budou do aplikace přidány detekce vybraných praktik a v souvislosti s tím také nové funkční prvky, které přispějí ke zvýšení udržitelnosti a škálovatelnosti aplikace při detekci špatných praktik a anti-vzorů z projektových dat, zejména pak z dat open-source projektů.

Práce je rozdělena do sedmi hlavních částí. První se zabývá projektovým řízením vývoje softwaru, jeho praktikami, vzory, anti-vzory, metodikami a podpůrnými ALM nástroji v procesu vývoje (kapitola 2). Další část popisuje analýzu projektových dat a framework SPADe (kapitola 3). V následující části dochází k analýze požadavků, tedy k popisu výchozí podoby aplikace SPAWn, a k návrhu vybraných detekcí špatných praktik (kapitola 4). Dále je proveden návrh potřebných funkčních rozšíření aplikace a jejich následná implementace (kapitola 5). V další části je popsána implementace navržených detekcí do aplikace (kapitola 6). Poté je proveden experiment, který se skládá z výběru sady open-source projektů, na které je vykonána detekce a následně je provedeno zhodnocení získaných výsledků (kapitola 7).

2 Řízení vývoje softwaru

Vývoj softwaru v dnešní době probíhá často formou projektu. Projekt je definovaný jako dočasné úsilí podniknuté k výrobě unikátního produktu, služby nebo výsledku s jednoznačným začátkem a koncem [13]. Projekt tedy můžeme chápat jako jednorázovou a jedinečnou sadu činností a procesů provedenou v ohraničeném čase za účelem dosažení určitého cíle. Činnosti v projektu jsou vzájemně provázané a jejich množství a složitost závisí na rozsahu každého projektu. Mezi činnosti prováděné v průběhu softwarového projektu patří nejen samotné psaní programového kódu, ale i řada podpůrných aktivit jako je tvorba specifikace či testování.

V této kapitole jsou uvedeny základní pojmy z oblasti projektového řízení, jeho praktik a často využívané metodiky k řízení projektu vývoje softwaru. V druhé části kapitoly jsou představeny podpůrné nástroje a jejich využití v projektu.

2.1 Projektové řízení

Nedílnou součástí realizace projektu je projektové řízení. Jedná se o organizované úsilí, jehož cílem je zajistit efektivní řízení sady činností tak, aby bylo dosaženo definovaného cíle při dodržení omezujících podmínek v podobě času, nákladů a požadované kvality [16]. Tyto podmínky jsou vzájemně provázané a nedodržení jedné z nich může vážně ohrozit i zbývající dvě. Například dodání produktu v nižší než požadované kvalitě může znamenat větší časové i finanční náklady na následnou úpravu či opravu. Tento fenomén je známý také jako projektový trojúhelník či projektový trojimperativ a je znázorněn na obrázku 2.1.

Odpovědnost za řízení projektu a jeho úspěšné dokončení přebírá projektový manažer. Jeho úkolem je řídit a sledovat průběh projektu ve všech fázích, zajišťovat potřebné zdroje, udržovat komunikaci se zákazníky a členy týmu a řešit otázky ohledně nákladů, rozpočtu, času, změn, kvality produktu a spokojenosti zákazníků i členů týmu [1]. Výčet aktivit a odpovědností projektového manažera se může zásadně lišit dle povahy a rozsahu projektu.

Za úspěšné dokončení projektu je považován stav, kdy dojde k dosažení stanovených cílů a naplnění požadavků stakeholderů, tedy osob zainteresovaných do projektu (především zákazníků) [27].

Obrázek 2.1: Projektový trojimperativ [14]

2.1.1 Praktiky a vzory

V projektovém řízení se často objevují opakované postupy známé jako praktiky. Jedná se o prvky kumulativní znalostní báze dílčích technik a postupů v oboru, ale také poznatky vycházející ze zkušeností každého manažera. Praktiky se mohou týkat různých oblastí řízení od personálních záležitostí až po oblast komunikace či plánování.

Dobré praktiky projektového řízení představují techniky, metody nebo procesy, které jsou považovány za efektivní při použití za konkrétního stavu nebo za určité okolnosti [12]. Využití dobrých praktik spočívá v aplikaci známých postupů, které dříve přispěly k úspěšnému dokončení projektu nebo jeho části, a tudíž je vhodné je znovu využít. Příkladem dobré praktiky může být využití retrospektivy nebo post-mortem analýzy pro získávání zpětné vazby o průběhu iterace nebo celého projektu. Dalším příkladem může být rozložení komplexních a náročných úkolů do více menších, které jsou lépe zvládnutelné.

Opačným případem dobrých praktik jsou špatné praktiky. Jedná se o opakovatelné postupy nevhodné pro kontext nebo mající negativní dopad na průběh projektu a jeho úspěšné dokončení. Těmto postupům je vhodné se při projektovém řízení vyhnout, a zamezit tak nežádoucím následkům. Mezi špatné praktiky je možné zařadit například neúplné či chybějící plánování nebo nedostatečnou komunikaci [25].

Anti-vzory a vzory se od dobrých či špatných praktik v zásadě neliší. Anti-vzory můžeme chápat jakožto podmnožinu špatných praktik, vzory pak jako podmnožinu dobrých praktik. Jedná se o známá a opakovaně se objevující dobrá či špatná řešení nějakého daného problému. Anti-vzory mohou

působit jako vhodná řešení, ale v daném kontextu problém pouze odsouvají, obchází či přeměňují v jiný [6]. Těmto postupům se, stejně jako u špatných praktik, snažíme vyhnout, a zamezit tak nežádoucím následkům.

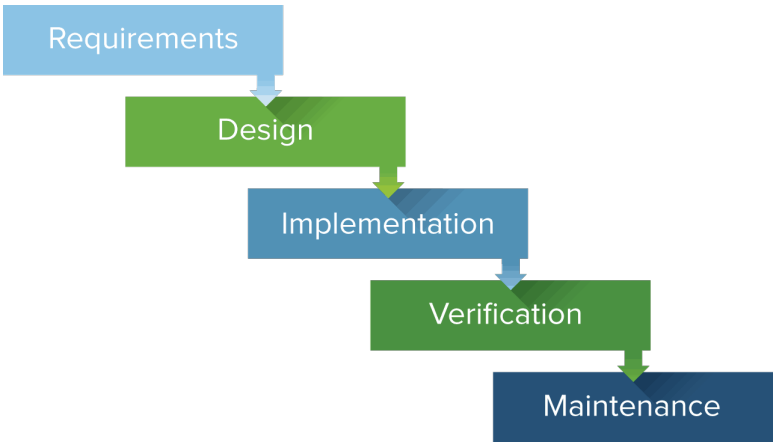
V softwarovém projektu můžeme rozlišit vzory a anti-vzory ve smyslu návrhových vzorů v programovém kódu. Příkladem může být vzor Singleton nebo anti-vzor zvaný Špagetový kód [17]. Dále se pak může jednat o vzory a anti-vzory v projektovém řízení, mezi které řadíme například vzor Take No Small Slips [8] nebo anti-vzor Analysis Paralysis [6].

2.1.2 Metodiky projektového řízení

V řízení projektu vývoje softwaru se často vyskytují takzvané metodiky vývoje softwaru. Jedná se o soustavu hlavních principů a procesů pro řízení projektu [7]. Metodika popisuje průběh procesu vývoje a do určité míry pokrývá zásadní aktivity v projektu. Volba metodiky je často závislá na povaze projektu a její výběr je jedním z prvních rozhodnutí projektového manažera, které bude mít zásadní vliv na průběh práce projektového týmu. Zde jsou uvedeny některé z často používaných metodik projektového řízení vývoje softwaru.

Sekvenční metodiky

Sekvenční metodiky se vyznačují rozdělením do samostatných fází, které na sebe lineárně navazují [7]. Jednotlivé fáze se navzájem neprolínají. Výhodou je jednoduchost a snadné plánování jednotlivých činností. Nevýhodou je pak nízká flexibilita a vysoké náklady na změny v požadavcích a opravu chyb. Z toho důvodu je kladen velký důraz na důkladnou analýzu požadavků v počáteční fázi projektu. Sekvenční metodiky jsou vhodné pro méně komplikované projekty s jasným cílem a daným postupem, ale i komplikovanější projekty, kdy je použití sekvenčního přístupu vynuceno okolnostmi. Nejvýznamnějším zástupcem sekvenčních metodik je Vodopádový model [23]. Příklad posloupnosti jednotlivých fází Vodopádového modelu je znázorněn na obrázku 2.2.

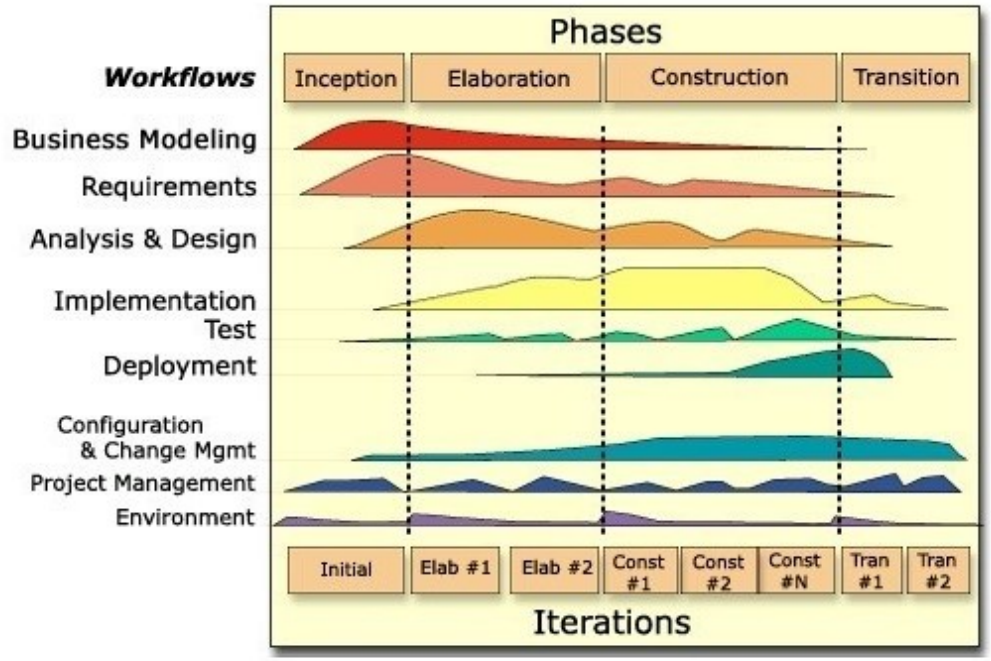


Obrázek 2.2: Vodopádový model [23]

Iterativní metodiky

Tato skupina metodik se vyznačuje využitím iterativního přístupu. Iterace je časový úsek zpravidla několik týdnů, který je možný chápat jako sekvenční přístup na menší škále. Každá iterace zahrnuje většinu z projektových aktivit, které jsou zastoupeny v různé míře. V počáteční fázi projektu je kladen větší důraz na sběr požadavků a důkladnou analýzu, zatímco v pozdější fázi je více času věnováno implementaci a testování. Každá iterace má definovaný soubor cílů a jejím výstupem je spustitelný program, který je vždy o krok blíže konečnému produktu.

Významným zástupcem iterativních metodik je Unified Process (UP), a především pak jeho verze Rational Unified Process (RUP) definovaná společností IBM. Metodika RUP je z důvodu robustnosti vhodná spíše pro větší a komplexnější projekty, ale ve zmenšené verzi může být použitá i pro méně náročné projekty [15]. Příklad zastoupení jednotlivých činností v různých fázích projektu znázorňuje obrázek 2.3.



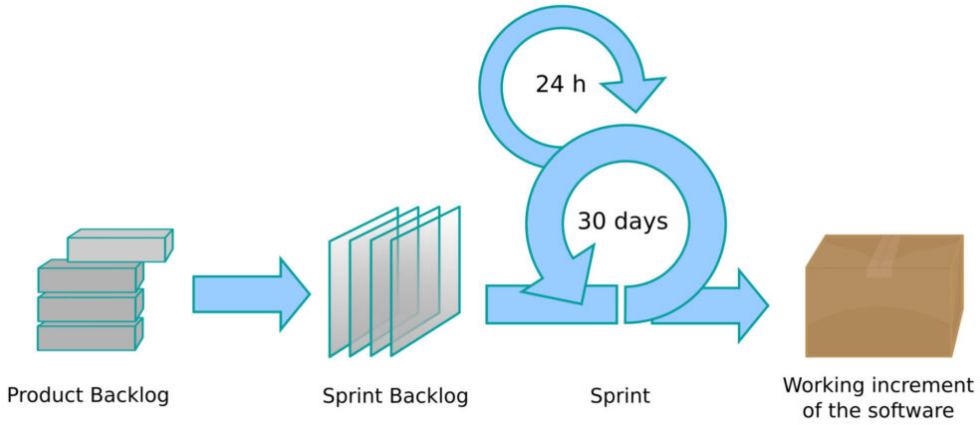
Obrázek 2.3: Zastoupení aktivit ve fázích RUP [15]

Agilní metodiky

Využití agilních metodik je v dnešní době velmi populárním přístupem v široké škále softwarových projektů. Jedná se o iterativní a inkrementální metodiky kladoucí důraz na flexibilitu a efektivní reakci na změny v požadavcích.

Agilní metodiky jsou založeny na Manifestu agilního vývoje softwaru, který shrnuje klíčové principy a zásady. Do popředí staví především častou a transparentní komunikaci se zákazníkem a pružnou reakci na změny v jeho potřebách a prioritách [2].

Mezi zástupce agilních metodik patří SCRUM, Extrémní programování a Test Driven Development. Na obrázku 2.4 je znázorněn příklad průběhu iterace v metodice SCRUM.



Obrázek 2.4: Průběh iterace v metodice SCRUM [10]

2.2 ALM nástroje a projektová data

Pro zajištění efektivního vývoje softwaru je vhodné, a při větších projektech i nezbytné, využívat podpůrné Application Lifecycle Management (ALM) nástroje. Tyto nástroje usnadní řadu činností a procesů ve všech fázích projektu, jako je například komunikace a spolupráce, distribuce a uchovávání informací, zálohování a verzování zdrojového kódu, podpora řízení a další.

Tato podkapitola popisuje vybrané skupiny ALM nástrojů, jejich použití v projektu vývoje softwaru a povahu dat, které tyto nástroje uchovávají.

2.2.1 Nástroje pro správu verzí

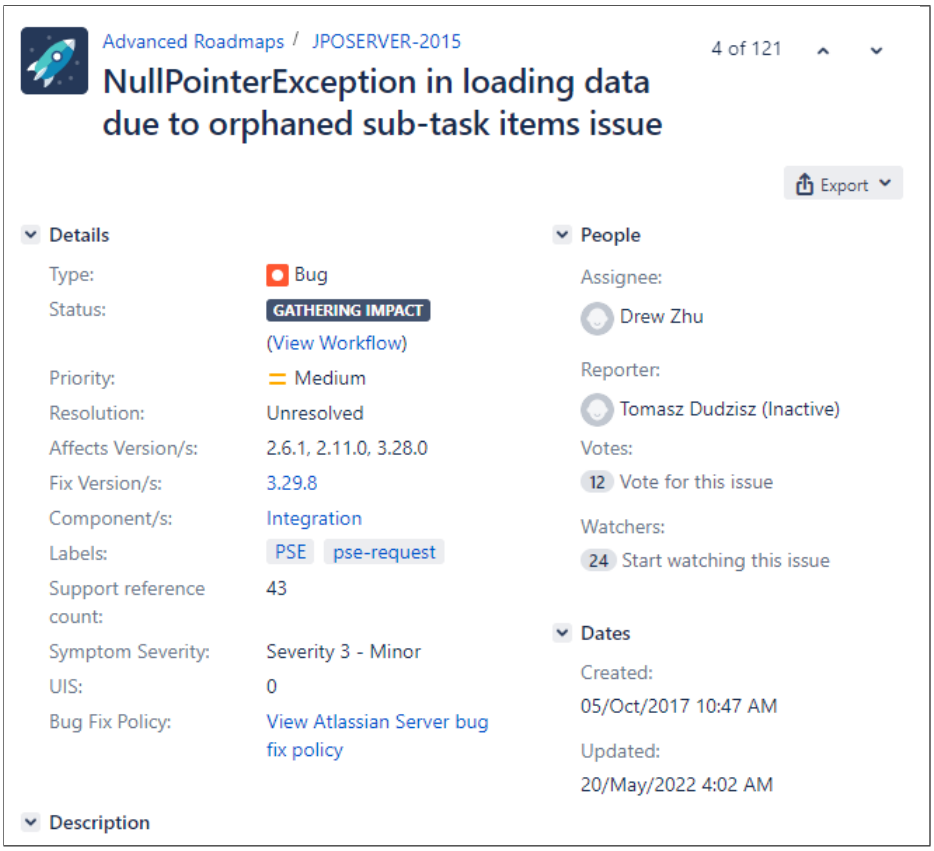
Nástroje pro správu verzí, nebo také Version Control Systems (VCS), zaznamenávají soubory v určitých fázích změny. K jednotlivým souborům ukládají informace o změnách, konkrétně kdo a kdy změnu provedl a o jakou změnu se jedná. Ukládání verzí poskytuje možnost vrácení se k původním verzím souborů. Kromě toho jsou soubory zálohované a je možné sledovat celkový pokrok a jednotlivé změny. V projektech vývoje softwaru slouží především ke zdrojovým kódům a dalším částem implementace, jako jsou například konfigurace nebo programová dokumentace. Mezi nástroje pro správu verzí řadíme například Git[1] nebo TortoiseSVN[2].

[1] https://git-scm.com
[2] https://www.tortoisesvn.net

2.2.2 Nástroje pro správu změn a požadavků

Nástroje pro správu změn a požadavků, známé také jako nástroje pro správu ticketů či bug-tracking nástroje, umožňují spravovat a udržovat přehled o jednotlivých úkolech a úlohách. V těchto nástrojích se často setkáváme s pojmem issue. Jedná se o konkrétní požadavek na změnu, úpravu či opravu. Issue obsahuje informace o autorovi (tzv. reporter) a popis problému nebo požadavku. Často je uvedena priorita úlohy, aktuální stav a další nezbytné či doplňující informace různého charakteru.

Mezi nástroje pro správu změn a požadavků řadíme například nástroje Atlassian Jira[3], Mozilla Bugzilla[4] nebo Redmine[5]. Na obrázku 2.5 je uvedený příklad issue v nástroji Atlassian Jira.



Obrázek 2.5: Issue v nástroji Atlassian Jira

[3]https://www.atlassian.com/software/jira
[4]https://www.bugzilla.org
[5]https://www.redmine.org

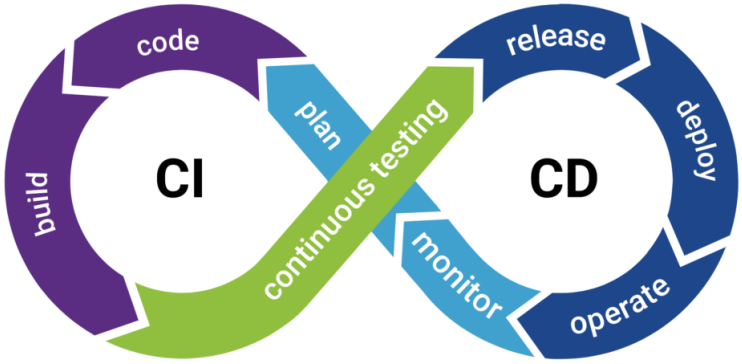
2.2.3 Nástroje pro správu nasazení

U produktů se serverovou částí může být nasazení nové verze do běhového (produkčního, testovacího apod.) prostředí podpořeno nástroji pro správu nasazení. Jejich využití spočívá v automatizaci a pravidelném sledování. Tento proces probíhá ve dvou fázích:

- **Continuous Integration (CI)** – Zahrnuje například změnu v kódu, opravení chyby či přidání nové funkcionality. Změna je často ověřována automatizovaně. Tím je možné odhalit chyby ještě před samotným nasazením do produkce.
- **Continuous Deployment (CD)** – Jedná se o automatizované testy a následné nasazení softwaru do produkce.

Aktivity probíhající v těchto fázích jsou znázorněny na obrázku 2.6.

V nástrojích pro správu nasazení se zaznamenávají informace o průběhu a výsledcích jednotlivých činností. Mezi tyto nástroje je možné zařadit například Jenkins[6] nebo CircleCI[7]. Některé nástroje mohou obsahovat CI/CD jako jednu z komponent. Takovým nástrojem je například GitLab[8].



Obrázek 2.6: Proces CI a CD [22]

2.2.4 Nástroje pro komunikaci

Komunikace je bezesporu klíčovou součástí každého projektu. Dobře zvládnutá komunikace umožní členům týmu spolupracovat, aniž by se museli fyzicky nacházet na stejném místě. Díky efektivní komunikaci je možné správně pochopit potřeby a požadavky zákazníků. V mnoha případech závisí kvalita

[6]https://www.jenkins.io
[7]https://www.circleci.com
[8]https://www.gitlab.com

všech výstupů projektu, a tedy i jeho celková úspešnost, právě na efektivitě komunikace. Velkou měrou ke zvládnutí komunikace v projektu přispívají komunikační nástroje. Tyto nástroje zajišťují komunikaci v různých podobách, od emailů až po videokonference. V řadě nástrojů je také možné již proběhlou komunikaci zpětně dohledat. Mezi nástroje pro komunikaci patří například Microsoft Teams[9], Skype[10] a Slack[11].

2.2.5 Další nástroje

Výčet kategorií ALM nástrojů není vyčerpávající. Mezi další kategorie patří například nástroje pro ověřování kvality (například Selenium[12]) nebo nástroje znalostního managementu (například Atlassian Confluence[13]). Také je nutno zmínit, že některé nástroje zastanou funkci všech, nebo alespoň několika kategorií zároveň. Takovým nástrojem je kupříkladu GitHub[14], který kromě verzování kódu, může sloužit i jako nástroj pro správu změn a požadavků.

[9]https://www.microsoft.com/microsoft-teams
[10]https://www.skype.com
[11]https://www.slack.com
[12]https://www.selenium.dev
[13]https://www.atlassian.com/software/confluence
[14]https://github.com

3 Analýzy projektových dat

V průběhu projektu se v podpůrných ALM nástrojích nashromáždí velké množství dat. To otevírá potenciál pro jejich další využití, tedy získání druhotných informací použitelných už za běhu zkoumaného projektu nebo po jeho skončení. Analýza dat umožní porozumět problémům a jejich příčinám nebo predikovat další vývoj.

Je však nutné zmínit, že výsledky analýz projektových dat často naráží na určitá úskalí v podobě zkreslení a neúplnosti zkoumaných dat. Získaná projektová data nemusí vždy plně odpovídat realitě. To může být způsobeno nedůsledným využíváním ALM nástrojů uživateli. Tento problém může pramenit z nedostatečné disciplíny nebo nevhodného výběru používaných ALM nástrojů, které ne zcela vyhovují potřebám vývojářů. Dalším důvodem může být částečná absence využití nástroje v průběhu projektu, čímž může dojít k vynechání důležitých informací, které nepůjde z ostatních dat odvodit [20]. Z těchto důvodů je pro správnou interpretaci získaných výsledků analýzy nutné znát kontext zkoumaného projektu.

Navíc každý nástroj uchovává svá data v různé podobě a struktuře a data jednotlivých projektů mohou být různě přístupná (veřejná, po přihlášení, jen pro přístup z vnitřku organizace, atd.).

Tato kapitola popisuje dělení projektů podle dostupnosti jejich dat, využití analýzy projektových dat a nástroje pro její provedení. Dále představí framework SPADe a jeho architekturu.

3.1 Dostupnost projektových dat

Podle přístupnosti projektových dat je možné rozdělit projekty do několika skupin.

Komerční projekty

Komerční projekty probíhají uvnitř softwarových společností. Projektová data u komerčních projektů jsou přístupná pouze v rámci pracovní skupiny, týmu či firmy. Jejich přístupnost může být dále interně škálovatelná dle jednotlivých rolí. Důvodem pro tuto omezenost přístupu může být bezpečnost či ochrana konkurenční výhody produktu.

Open-source projekty

Jedná se o projekty s volně přístupným zdrojovým kódem. Do vývoje je možné volně přispívat nebo využít zdrojový kód pro vlastní potřeby. Ačkoliv je vyvíjený produkt volně přístupný, dostupnost některých projektových dat je často omezená. Ve většině případů je dostupný zdrojový kód a informace o provedených změnách, jednotlivých úkolech a proběhnutých diskuzích. Naopak některá data, například o průběhu testování, analýzách požadavků nebo nasazení aplikace, často dostupná nejsou.

Studentské projekty

Studentské projekty probíhají v rámci výuky či výzkumu na dané instituci, zejména univerzitě. Data z těchto projektů jsou často přístupná uvnitř instituce pro další účely. Především se jedná o použití dat v navazujících výzkumech, využití při výuce nebo čistě o archivaci. Díky jejich účelu se studentské projekty obvykle liší od projektů komerčních a jejich využití je často vnímáno jako nereprezentativní vzorek.

3.2 Využití analýzy dat

Analýzou projektových dat lze získat řadu dalších informací, které mohou mít různorodé využití. Během projektu je možné zajistit informace týkající se dosavadního průběhu, které pak mohou sloužit projektovému manažerovi jako podklad pro další rozhodování. Zároveň mohou usnadnit sledování procesu, kontrolu shody s jeho zamýšlenou podobou a zachycení případných problémů. Pokud se v projektu některý problém vyskytne, analýzou dat je možné zjistit jeho příčinu nebo identifikovat potenciálně rizikové momenty a faktory, které k němu přispěly. Problémy je zároveň možné do určité míry predikovat a zavést opatření pro zamezení nebo zmírnění jejich následků.

Data hrají významnou roli i po skončení projektu. Analýzou nashromážděných dat lze provést různorodé bilance, a získat tak informace využitelné v dalších projektech. Zároveň je umožněno projekty mezi sebou porovnávat. V případě neúspěchu projektu lze využít takzvané post-mortem analýzy a zjistit příčinu, která vedla k tomu, že se projekt stal neúspěšným.

Provádět analýzy projektových dat je možné přímo v ALM nástrojích. Některé ALM nástroje obsahují analytické funkce, které poskytují důležité informace o stavu a průběhu projektu. Takovým nástrojem je kupříkladu GitHub. Dále se může jednat o externí nástroje a služby, pomocí kterých je možné se k datům z ALM nástrojů přímo připojit nebo z nich data převést do

různých forem exportu, na kterých je následně umožněno provádět analýzy. Mezi takové nástroje patří například PowerBI nebo Microsoft Excel.

V akademické sféře se můžeme setkat s různými projekty vyvíjející nástroje pro analýzy projektových dat. Příkladem může být projekt CHAOSS [18], poskytující aplikaci pro analýzu udržitelnosti open-source projektů, nebo nástroj YOSHI [24], který se zabývá zejména organizačními aspekty v projektech.

3.3 Framework SPADe

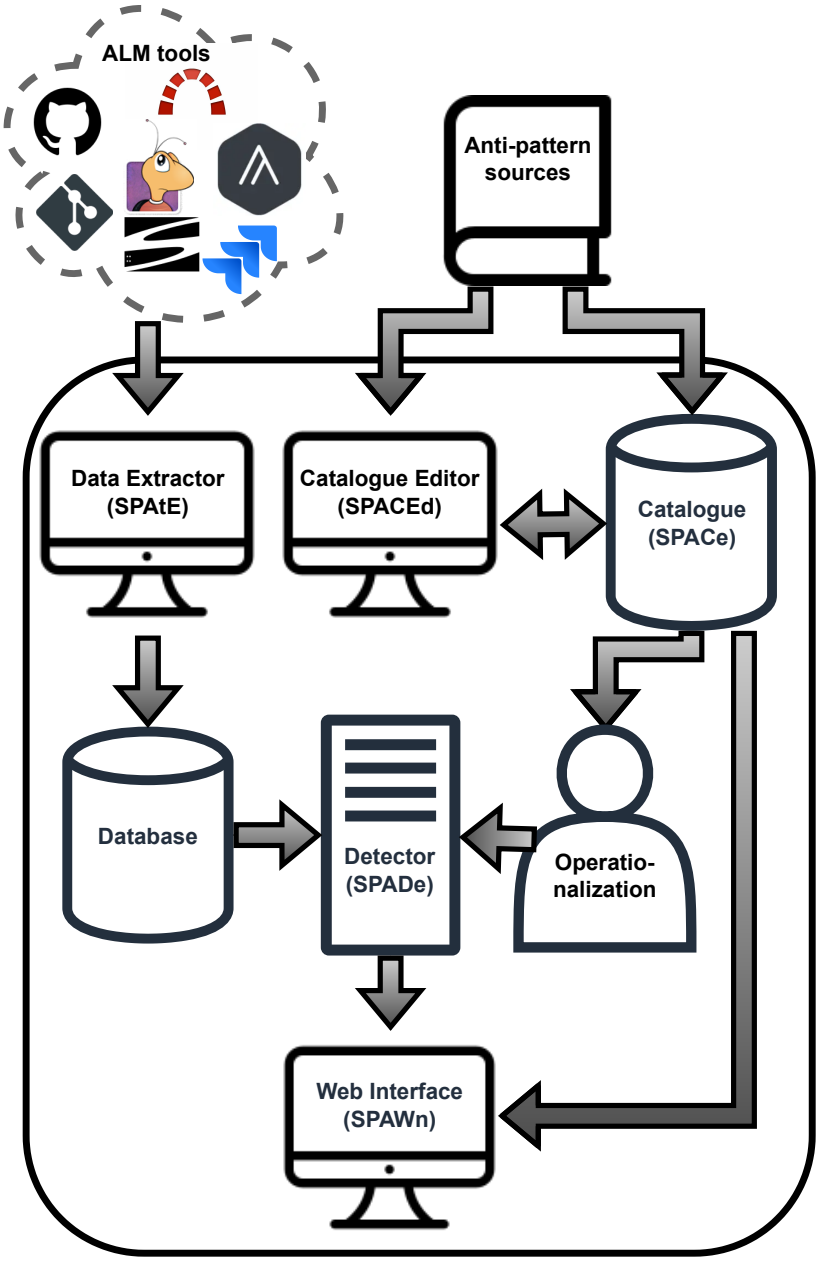
Během projektu vývoje softwaru se projektový manažer snaží vyhnout špatným praktikám a anti-vzorům, aby tak předešel případnému snížení pravděpodobnosti úspěšného dokončení projektu. Některé špatné praktiky však na první pohled nemusí být zřejmé a k jejich odhalení může dojít až v případě, kdy už významně negativně ovlivnily průběh projektu. Tento problém by mohla řešit automatická detekce špatných praktik a anti-vzorů už v jejich počáteční fázi. Díky tomu by měl projektový manažer prostor adekvátně zareagovat.

V současné době neexistuje komerční nástroj, který by v projektových datech prováděl automatickou detekci špatných praktik nebo anti-vzorů nezávisle na použitém ALM nástroji. Tento problém je předmětem projektu Katedry informatiky a výpočetní techniky (KIV) Fakulty aplikovaných věd (FAV) Západočeské univerzity v Plzni (ZČU), kde je vyvíjen framework Software Anti-pattern Detector (SPADe).

Hlavní funkčností frameworku SPADe je těžba dat ze softwarových projektů uložených v různých ALM nástrojích a jejich transformace do unifikované struktury v datovém skladu. Nad naplněným datovým skladem je následně umožněno provádět datové analýzy zjišťující přítomnost událostí, vzorců chování, konceptů a metod. Zvláštní pozornost je věnována zejména přítomnosti špatných praktik a anti-vzorů [20].

Framework SPADe se skládá z několika samostatných částí a schéma jeho architektury je znázorněno na obrázku 3.1.

První z částí nástroje je soubor datových pump (Data Extractor). Jeho úkolem je dolovat data z vybraných ALM nástrojů a následně je uložit do jednotné struktury. Dolování probíhá za pomocí Extract-Transform-Load (ETL) procesu [11]. Pro dolování dat je využito rozhraní daných ALM nástrojů, které však není jednotné. Z toho důvodu je pro každý nástroj nadefinována vlastní ETL datová pumpa. V současné chvíli je umožněno získávat



Obrázek 3.1: Architektura frameworku SPADe

data z nástrojů Assembla[1], Mozilla Bugzilla, Git, GitHub, Atlassian Jira, Redmine, Apache SVN[2] [19]. Existují v zásadě čtyři hlavní metody pro získávání dat z nástrojů. Prvním je přímý přístup do databáze. Další a nejčastěji používaný je přístup pomocí rozhraní daného nástroje. Dále se může jednat o export do externího souboru a jeho následné rozparsování. Posledním přístupem je parsování jednotlivých stránek uživatelského rozhraní nástroje (tzv. web-crawling) [20].

Natěžená projektová data jsou uložena do jednotné struktury v datovém skladu (Database). Datový sklad je zvláštní typ databáze určený primárně pro analýzy dat, zejména v rámci Business Intelligence, tedy analýzy dat sloužící jako podklady pro manažerské rozhodování [9]. V tuto chvíli je datový sklad SPADe technicky realizovaný standardní relační databází.

Další částí frameworku je online katalog (Catalogue) v nástroji GitHub sdružující definice fenoménů určených k detekci v projektových datech, tedy špatných praktik a anti-vzorů. Pro každý fenomén je uveden stručný popis, výčet jeho příznaků a možných následků v případě, že se v projektu vyskytne. Ke katalogu je dále připojená aplikace pro jeho správu (Catalogue Editor), která umožňuje snadné přidávání a editaci popisů fenoménů v předem připravených šablonách tak, aby byla zachována celková konzistence katalogu.

Nedílnou součástí pro detekci špatných praktik a anti-vzorů je proces operacionalizace (Operationalization). Operacionalizace znamená přeměnu nebo transformaci abstraktního jevu na měřitelné veličiny [5]. Jinými slovy se jedná o detailní postup zachycení určitého fenoménu pomocí jednoho nebo více ukazatelů měřitelných nad natěženými projektovými daty.

Samotnou detekci fenoménů zajišťuje detektor (Detector) představující hlavní aplikační logiku frameworku SPADe. Dochází zde k analýzám projektových dat získaných pomocí dotazů z datového skladu. Detekce v datech je provedena na základě operacionalizovaných modelů daných špatných praktik nebo anti-vzorů.

Výsledky analýz jsou předány do aplikace představující webového rozhraní (Web Interface) pro jejich zobrazení uživateli. Toto webové rozhraní slouží pro přímou interakci s uživatelem. V této části může uživatel spouštět jednotlivé analýzy projektů a procházet získané výsledky. Zároveň si může zobrazit základní informace o projektech i detekovaných fenoménech.

[1] https://get.assembla.com
[2] https://subversion.apache.org

4 Aplikace SPAWn a nové detekce

V této kapitole bude nejprve zanalyzován pro tuto práci výchozí stav aplikace SPADe Web Interface (SPAWn), její vlastnosti a implementace. Zároveň bude popsána struktura datového skladu, který aplikace využívá. Poté budou vybrány a navrženy modely detekcí špatných praktik pro jejich následnou implementaci do aplikace.

4.1 Výchozí stav aplikace SPAWn

SPAWn je webová aplikace vytvořená v rámci předešlé diplomové práce, kterou vypracoval Ondřej Váně [26]. Aplikace SPAWn v tuto chvíli zastává funkci webového rozhraní a rovněž i detektoru, tedy poskytuje detekce vybraných anti-vzorů nad projektovými daty uloženými v datovém skladu SPADe. Data jsou ze skladu získána dotazovacím jazykem Structured Query Language (SQL) a následně procedurálně vyhodnocena. Poté jsou výsledky detekce v jednotlivých projektech přehledně zobrazeny uživateli.

V této podkapitole bude popsána výchozí podoba aplikace SPAWn, která je výsledkem diplomové práce [26] ve verzi 1.0.0 a následné spolupráce autora s KIV ve verzi 1.1.0[1]. Budou popsány její funkční vlastnosti, struktura a implementace.

4.1.1 Funkční vlastnosti aplikace

Tato část popisuje klíčové funkční vlastnosti aplikace SPAWn ve výchozí verzi 1.1.0.

Čtení z datového skladu SPADe

Projektová data určená k detekci přítomnosti anti-vzorů se nachází v datovém skladu SPADe a jsou získávána pomocí standardních SQL dotazů. Pro zajištění přehlednosti a znovupoužitelnost jsou SQL dotazy uchovány v samostatných souborech, které jsou součástí výsledné aplikace. Při spuštění detekce dojde k načtení potřebných SQL dotazů, do kterých se vloží potřebné parametry.

[1] https://github.com/ReliSA/SPADe-Web-GUI/releases/tag/1.1.0

Výběr anti-vzorů a projektů pro analýzu

Všechny dostupné projekty uložené v datovém skladu SPADe a všechny operacionalizované anti-vzory jsou přehledně zobrazeny pomocí uživatelského rozhraní. To poskytne uživateli výběr kombinace projektů a anti-vzorů, které chce detekovat.

Nastavení prahových hodnot

Aplikace poskytuje konfiguraci prahových hodnot pro detekce jednotlivých anti-vzorů. Nastavit prahové hodnoty je možné v aplikaci pomocí uživatelského rozhraní.

Přehledné zobrazení výsledků

Získané výsledky detekcí anti-vzorů pro jednotlivé projekty jsou zaneseny do přehledné tabulky a zobrazeny uživateli. U každého výsledku jsou zobrazeny podrobnější informace, na základě kterých byl anti-vzor detekován. Zobrazení výsledků pro všechny projekty v jedné tabulce umožní uživateli mezi sebou tyto projekty porovnávat.

Další funkce

Aplikace umožňuje přejít na detail anti-vzorů i projektů pro zobrazení jejich základních charakteristik. U anti-vzorů obsažených v katalogu je také zobrazený odkaz pro přechod na jejich stránku v online katalogu, který je součástí frameworku SPADe.

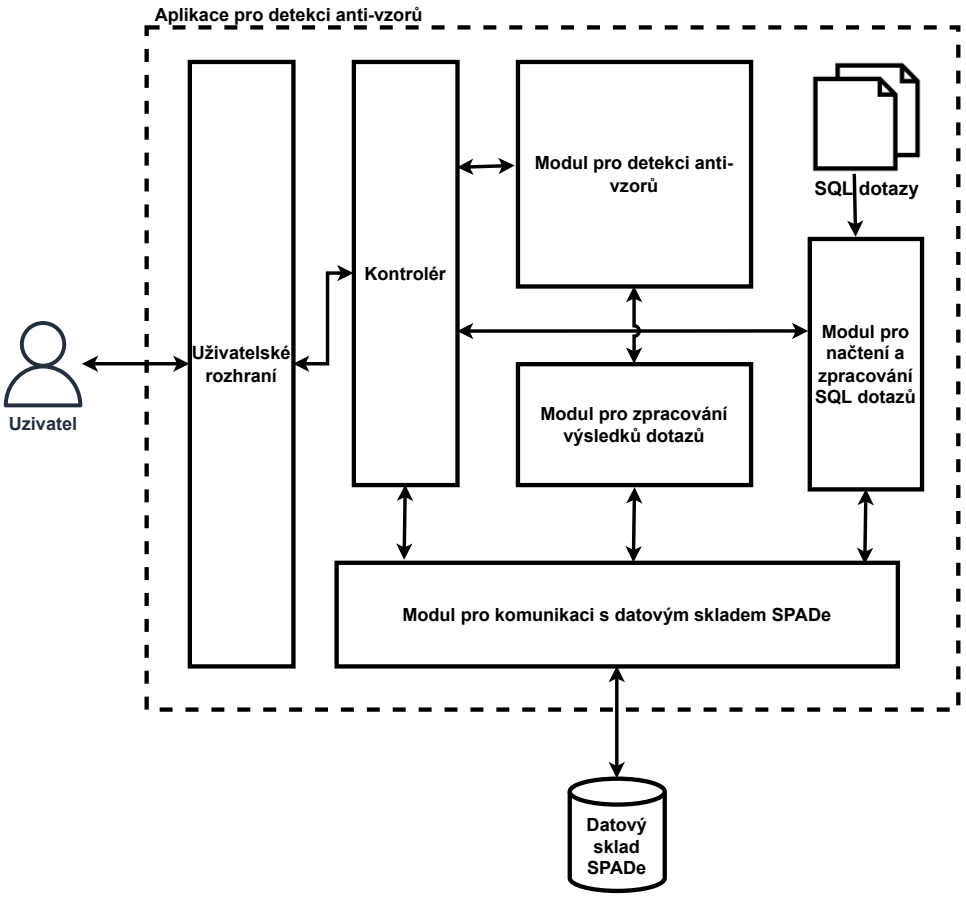
4.1.2 Struktura aplikace

Aplikace je rozdělena do šesti hlavních částí a její celková struktura je znázorněna na obrázku 4.1.

Interakce s uživatelem probíhá prostřednictvím uživatelského rozhraní, ve kterém se spouští detekce nad vybranou množinou anti-vzorů a projektů. Jednotlivé události vyvolané uživatelem obsluhuje tzv. kontrolér, který zajišťuje komunikaci s ostatními částmi aplikace.

Pro provedení detekcí jsou nezbytná data z datového skladu SPADe. Data jsou získána za pomoci SQL dotazů, které jsou uchovány v samostatných souborech. Do dotazů jsou nejprve vloženy potřebné parametry (prahové hodnoty) a následně dojde k provedení dotazů nad datovým skladem. Výsledky dotazů jsou transformovány do formátu, který je v aplikaci dále využíván.

Získaná data jsou procedurálně zanalyzována pro zjištění přítomnosti vybraných anti-vzorů. Výsledky detekce jsou předány kontroléru, který je připravuje k zobrazení uživateli v uživatelském rozhraní.



Obrázek 4.1: Výchozí struktura aplikace SPAWn [26]

4.1.3 Implementace aplikace

Aplikace je implementována monolitickou architekturou jako webová aplikace. Pro vývoj byla použita Java 11 s využitím open-source frameworku Spring Boot 2.4.4. Uživatelské rozhraní je implementováno pomocí šablonovacího systému Thymeleaf. Datový sklad je uložený v databázi MySQL 8.0.29 a pro komunikaci s ním je využito standardního Java Database Connectivity (JDBC) konektoru.

Struktura projektu

V tabulce 4.1 je zobrazena struktura projektu aplikace SPAWn. Pro zvýšení přehlednosti a úspory místa v tabulce je znakem `*` nahrazena cesta `src/main`

a znakem `#` je nahrazena složka `resources`.

Složka/Soubor	*Obsah*
`*/java`	zdrojové kódy aplikace
`*/#/application.properties`	nastavení aplikace
`*/webapp/queries`	definice dotazů pro anti-vzory
`*/webapp/WEB-INF/templates`	HTML šablony uživatelského rozhraní
`pom.xml`	soubor pro sestavení aplikace
`db_dump.sql`	soubor pro obnovu datového skladu
`Dockerfile`	soubor pro spuštění aplikace v Dockeru
`docker-compose.yml`	soubor pro definici služeb v Dockeru

Tabulka 4.1: Výchozí struktura projektu aplikace [26]

Balíky tříd

Třídy aplikace jsou členěny do balíků a jednotlivé balíky jsou umístěny v nadbalíku `cz.zcu.fav.kiv.antipatterndetectionapp`. V tomto nadbalíku je také umístěna hlavní třída `AntiPatternDetectionAppApplication` zajišťující spuštění celé aplikace. Projekt je tvořený celkem ze sedmi balíků.

- `controller` – Balík obsahující třídu pro zpracování požadavků uživatelů a následné zobrazení dat do uživatelského rozhraní.
- `detecting` – Balík obsahující třídy pro připojení k datovému skladu a detekci jednotlivých anti-vzorů.
- `model` – Balík obsahující modelové třídy pro předávání dat v rámci aplikace.
- `repository` – Balík obsahující třídy pro inicializaci jednotlivých detekcí a načítání SQL dotazů ze souborů.
- `service` – Balík obsahující služby pro práci s instancemi projektů a anti-vzorů.
- `spring` – Balík obsahující třídy pro konfiguraci frameworku Spring.
- `utils` – Balík obsahující třídu s pomocnými metodami.

Implementované detekce

Aplikace implementuje detekce pro sedm anti-vzorů. Konkrétně se jedná o tyto anti-vzory:

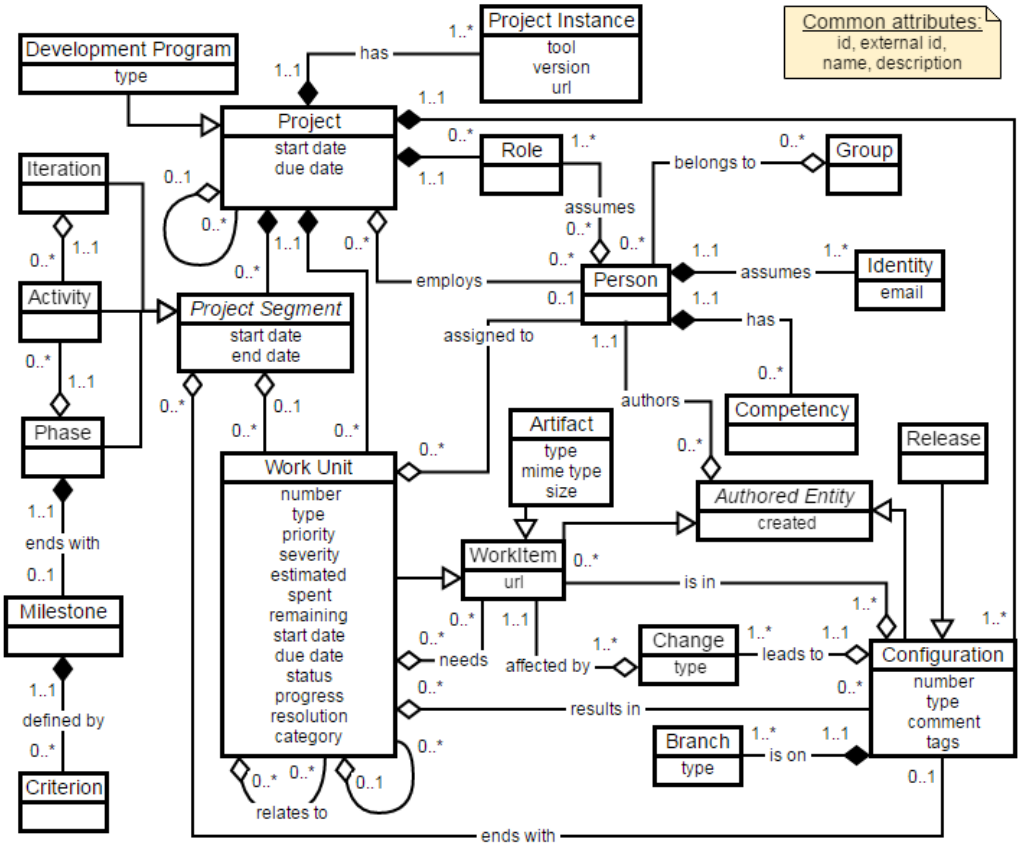
- Business As Usual,
- Long Or Non-Existant Feedback Loops,
- Ninety-Ninety Rule,
- Road To Nowhere,
- Specify Nothing,
- Too Long Sprint,
- Varying Sprint Length.

4.1.4 Struktura datového skladu

Základní strukturu datového skladu SPADe zachycuje zjednodušený doménový model na obrázku 4.2. Následující seznam popisuje méně intuitivní entity a entity relevantní pro tuto práci. Mezi tyto patří:

- **Artifact** – Entita obsahující důležité informace o artefaktech z nástrojů pro správu verzí a nástrojů pro správu změn a požadavků. Jedná se o jednotlivé soubory, složky, wiki stránky a přílohy.
- **Configuration** – Entita obsahující informace o stavu projektu v daném časovém bodě. Jedná se o souhrn změn, primárně artefaktů, úkolů a jiných předchozích konfigurací (např. přidání komentáře ke commitu) oproti předchozímu stavu.
- **Identity** – Entita představující účet uživatele použitý v daném ALM nástroji. Jedna osoba může mít více identit i v rámci jednoho nástroje.
- **Person** – Entita reprezentující osobu podílející se na projektu. Typicky se jedná o člena vývojového týmu, manažera a další osoby s přístupem do ALM nástrojů.
- **Project Instance** – Entita reprezentující instanci projektu v ALM nástroji. Projekt může mít více instancí, a to v různých nástrojích.
- **Work Item** – Rodičovská entita pro entity Artifact, Configuration a Work Unit.

- **Work Unit** – Entita obsahující informace o jednotlivých úkolech v projektu [19].



Obrázek 4.2: Doménový model datového skladu SPADe [20]

Další část datového skladu tvoří sada databázových pohledů vytvořených v rámci předešlé diplomové práce Patrika Bezděka [4]. Datový sklad SPADe je tvořen celkem ze 47 tabulek a informace o jednotlivých entitách jsou často z různých důvodů rozděleny v několika tabulkách. Databázové pohledy sdružují relevantní data o nějaké entitě z více tabulek do jedné logické struktury. V datovém skladu SPADe se nachází celkem osm pohledů pro koncepty jako jsou artefakt, úkol, commit, sada změn, osoba atd.

4.2 Výběr špatných praktik a návrh detekce

Špatné praktiky jsou nadmnožinou anti-vzoru, tzn. všechny anti-vzory jsou zároveň špatnými praktikami. Špatná praktika může být i součástí anti-vzoru. Může se jednat o podsadu jeho indikátorů, které nemusí nutně zna-

menat výskyt samotného anti-vzoru, ale už přítomnost těchto indikátorů má na projekt negativní vliv a je vhodné na ni upozornit.

V tomto duchu, výběr špatný praktik k detekci v této práci byl proveden na základě výběru a modelování celých anti-vzorů, které ale nebyly nutně modelovány v celém svém rozsahu, ale mohou na něj být v budoucnu rozšířeny. Ač tedy navržené modely fakticky detekují špatné praktiky, pro potřeby popisů budou dále označovány za anti-vzory.

Hlavním kritériem pro výběr anti-vzoru k detekci je možnost anti-vzor namodelovat na základě projektových dat. Některé dostupné anti-vzory z online katalogu (viz 3.3) mohou být příliš abstraktní, popřípadě mohou záviset na mnoha faktorech, o kterých nelze z projektových dat získat informace. Často se jedná o anti-vzory týkající se vlastností a dovedností osob zapojených do projektu, problémy personalistiky či jiných oddělení, jejichž aktivita není v ALM reflektována, nebo chování členů týmu v přímé osobní komunikaci.

Pokud anti-vzor splňuje výše zmíněné hlavní kritérium, je další podmínkou pro jeho výběr přítomnost projektových dat potřebných k jeho detekci v některém z ALM nástrojů. Absence dat může vznikat z několika důvodů. Jedním z nich může být nevyužití takového ALM nástroje, který by požadovaná data zachycoval (např. GitHub nemá koncept pro odhady pracnosti a vykazování stráveného času), popřípadě část ALM nástroje, která tato data zachycuje, není v projektu využívána. Dalším důvodem může být nekompetentní použití ALM, popř. špatné týmové konvence užívání ALM, nebo jejich nedodržování.

Důležitým kritériem je také přítomnost konkrétních projektových dat, potřebných k detekci daného anti-vzoru (např. znalostní báze projektu), v datovém skladu SPADe. Data se mohou nacházet v ALM nástroji (např. Atlassian Confluence), pro který není v nástroji SPADe nadefinována datová pumpa, popřípadě taková data pumpa vůbec netěží. V tom případě nebude v současné chvíli možné anti-vzor ani špatnou praktiku na těchto datech závislých detekovat.

Na základě analýzy dle zmíněných kritérií byla vybrána sada tří anti-vzorů k detekci. Jedná se o tyto anti-vzory:

- Unknown Poster,
- Bystander Apathy,
- Yet Another Programmer.

V dalších částech jsou vybrané anti-vzory popsány a je navržen model postupu jejich detekce nebo detekce špatných praktik ze kterých vycházejí.

Model detekce je vytvořen na základě dostupných dat v datovém skladě SPADe.

4.2.1 Unknown Poster

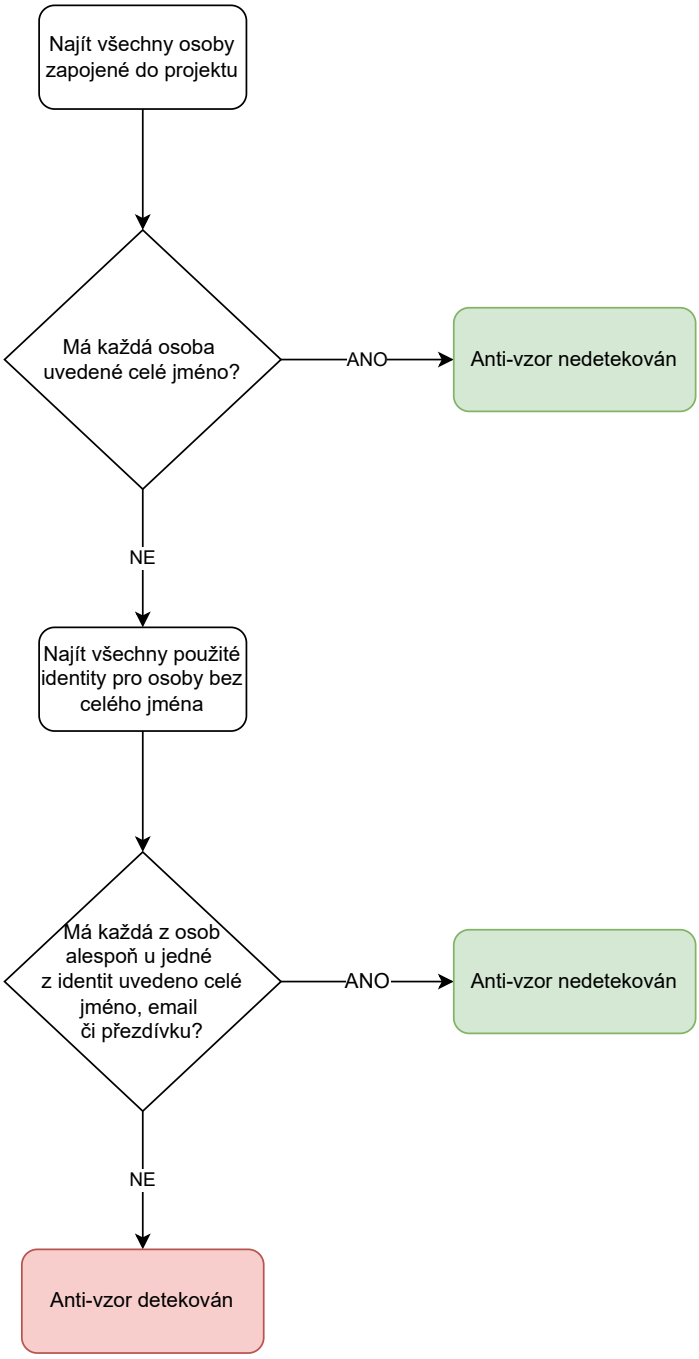
Popis

Tento anti-vzor se zabývá přítomností neznámého přispěvatele do projektu. U tohoto přispěvatele není možné určit jeho identitu. To může být způsobeno tím, že uživatel není přihlášený nebo v uživatelském účtu nemá správně vyplněné jméno, přezdívku nebo jiný údaj, podle kterého by jej bylo možné identifikovat. V některých případech k tomuto může dojít i cíleně, kdy uživatel nechce být spojený s určitým příspěvkem.

Přítomnost tohoto anti-vzoru s sebou nese určitá rizika. V případě potřeby není možné zjistit, kdo je za daný příspěvek zodpovědný a na koho se v případě potřeby obrátit. K zamezení výskytu neznámého přispěvatele by mohl sloužit zákaz příspěvků od nepřihlášených uživatelů nebo povinné vyplnění celého jména či emailu v uživatelských účtech [3].

Detekce

Detekce tohoto anti-vzoru bude probíhat formou kontroly údajů u osob a jimi použitých identit. V prvním kroku budou vybrány veškeré osoby zapojené do zkoumaného projektu. V případě, že bude mít každá osoba uvedené celé jméno, nedošlo tak k nalezení neznámého přispěvatele, tudíž anti-vzor není přítomen. V opačném případě se pro osoby bez uvedeného jména budou zkoumat jimi použité identity. Má-li osoba alespoň u jedné z identit uvedený kontaktní email, je možné ji prohlásit za identifikovatelnou. V opačném případě je potřeba zkontrolovat, zda je místo jména uvedena alespoň přezdívka, podle které by identifikace byla možná. Pokud v projektu existuje alespoň jedna osoba, kterou nepůjde identifikovat na základě těchto údajů, lze anti-vzor prohlásit za detekovaný. Model detekce je znázorněný stavovým diagramem na obrázku 4.3.



Obrázek 4.3: Stavový diagram detekce Unknown Poster

4.2.2 Bystander Apathy

Popis

Pojem Bystander Apathy, nebo také Bystander Effect, označuje psychologický jev, kdy jsou lidé neochotní nabídnout pomoc ostatním. V softwarovém projektu se tento jev může vyskytnout například v podobě otázky nebo žádosti o pomoc, které ale nikdo nevyhoví. To má za následek neefektivní práci žadatele, který na odpověď musí dlouho čekat, nebo se dlouho vypořádávat se samostatným řešením. Výsledkem může být příliš mnoho stráveného času nad problémem, který by mohl být snadno vyřešen za pomoci jiné osoby. Tento anti-vzor vychází ze špatné praktiky, která poukazuje na špatnou komunikaci a neefektivní nebo chybějící spolupráci v projektovém týmu [28].

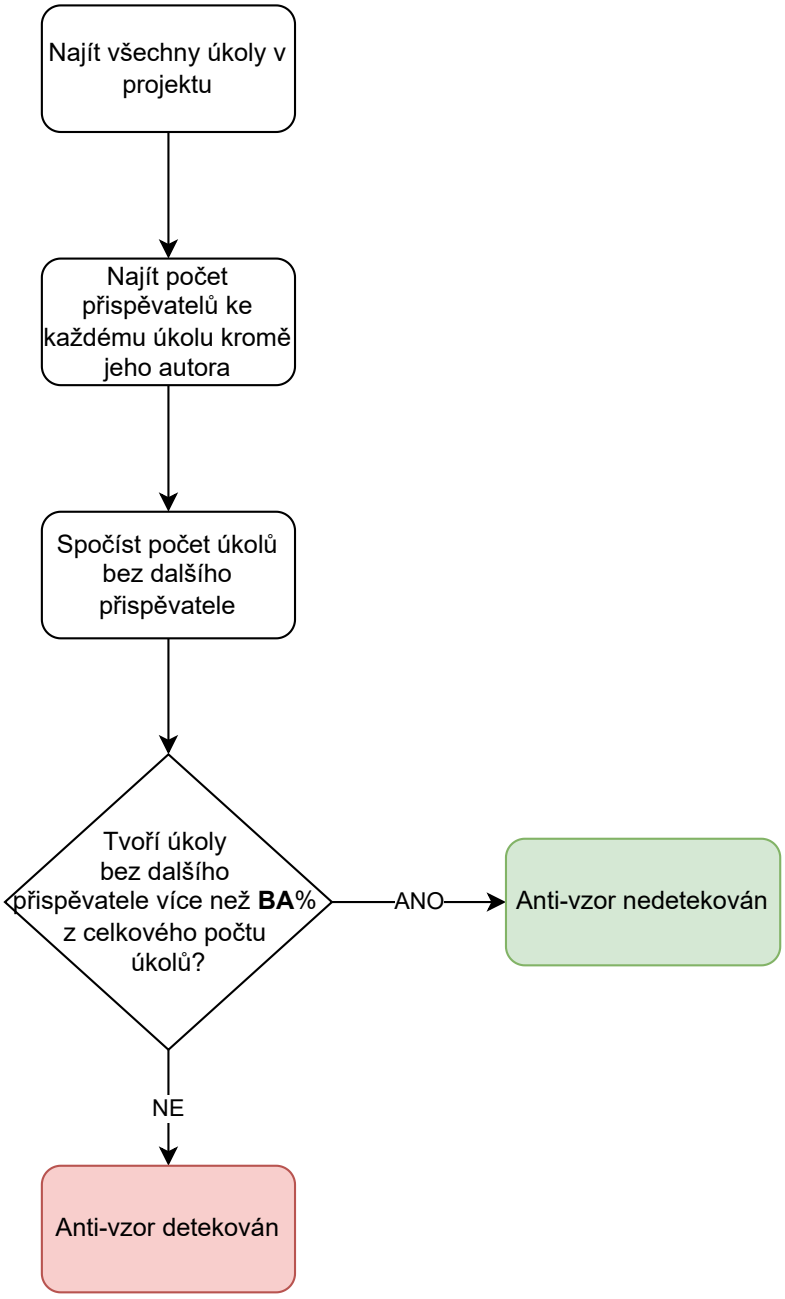
Detekce

Návrh detekce není omezený na přesnou shodu s definicí anti-vzoru, ale zaměří se na problematiku absence spolupráce obecně, tedy špatnou praktiku. Detekce cílí na vyhledání úkolů, u kterých žádná spolupráce neprobíhala, tedy tyto úkoly neobsahují žádného dalšího přispěvatele, kromě autora úkolu.

Nejprve budou vyhledány všechny úkoly ve zkoumaném projektu a ke každému úkolu bude nalezen počet všech přispěvatelů kromě samotného autora tohoto úkolu. V případě jediného přispěvatele je v úkolu nalezena absence spolupráce.

Pokud počet všech úkolů s absencí spolupráce bude tvořit významnou část z celkového počtu úkolů, bude anti-vzor prohlášen za detekovaný. Hranice pro detekci bude určena prahovou hodnotou `BA` představující procentuální část z celkového počtu úkolů, při jejíž překročení bude anti-vzor detekován. Postup detekce je znázorněný stavovým diagramem na obrázku 4.4.

Tento model detekce by bylo možné dále rozšířit o zkoumání časů a míry zapojení jednotlivých přispěvatelů do úkolu. Pokud by například jeden přispěvatel pracoval na úkolu průběžně po celou dobu, zatímco jiný přispěvatel by se do úkolu zapojil až na samém konci málo významným příspěvkem, není možné říci, že by v úkolu probíhala spolupráce. Hodnocení na základě časů a míry zapojení osob je však příliš komplexní a vyžaduje nutnost dobře znát kontext celého projektu i jednotlivých úkolů a z toho důvodu nebude do detekce zahrnuto.



Obrázek 4.4: Stavový diagram detekce Bystander Apathy

4.2.3 Yet Another Programmer

Popis

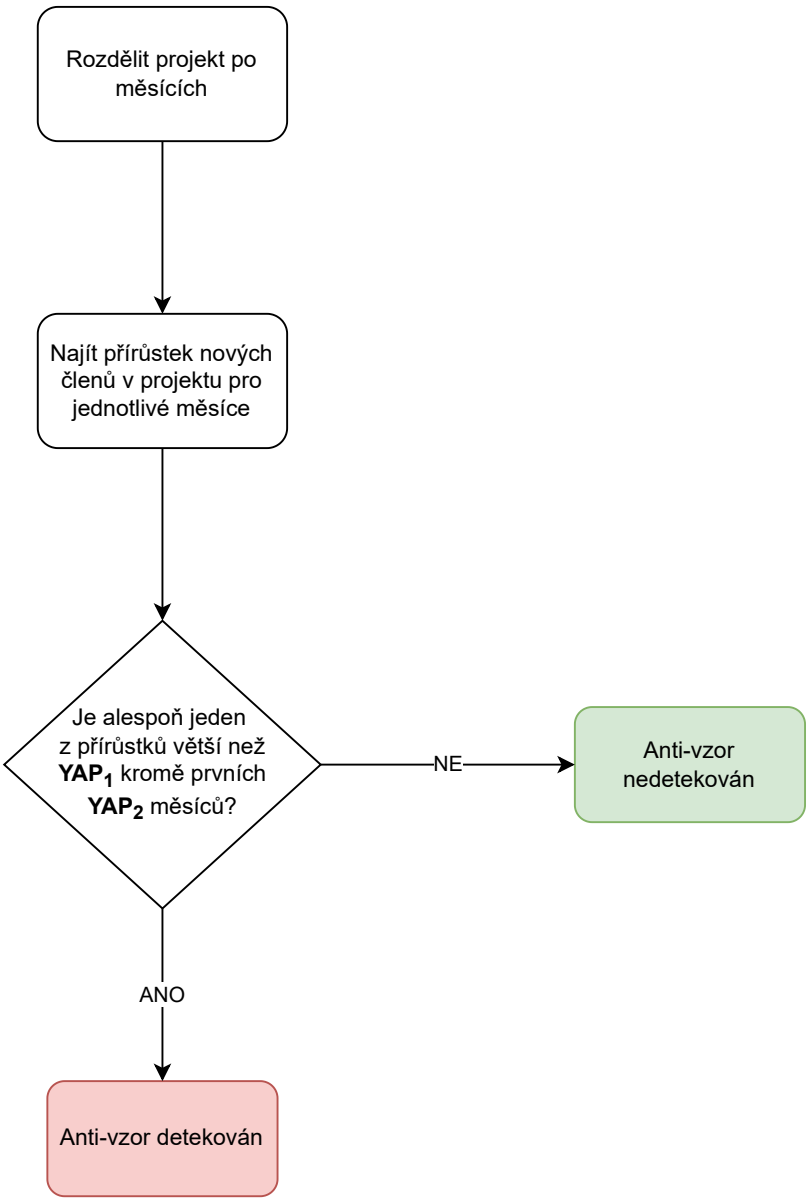
Tento anti-vzor popisuje situaci, kdy dochází k velkému zvýšení počtu nových vývojářů nebo výměně stávajících za běhu projektu. Tento jev se často vyskytuje zejména ke konci projektu, kdy má zapojení nových programátorů urychlit vývoj, a zajistit tak včasné dodání produktu. Ve skutečnosti má však často opačný efekt. Očekávání, že noví pracovníci urychlí proces může být chybné, jelikož zapojení nových vývojářů do rozběhnutého projektu zapříčiní dodatečné časové výdaje na jejich zaškolení a začlenění do týmu [29].

Detekce

Detekce anti-vzoru spočívá v kontrole počtu nových pracovníků v jednotlivých měsících. Nejprve dojde k rozdělení projektu na jednotlivé měsíce a následně bude získán počet nově zapojených pracovníků pro každý měsíc. Za datum zapojení každého pracovníka bude považován datum jeho prvního příspěvku do projektu. Pokud v některém z měsíců přesáhl počet nově příchozích pracovníků určitý počet, bude tento anti-vzor považován za detekovaný. Maximální počet nových pracovníků v jednom měsíci, aniž by se jednalo o anti-vzor, bude zadán jako prahová hodnota $\texttt{YAP}_1$.

Je ovšem nutné z detekce vyřadit počáteční fázi projektu, tedy část projektu, ve kterém se nachází první příspěvky od pracovníků, kteří s projektem začínali. Tato fáze projektu bude určená prahovou hodnotou $\texttt{YAP}_2$, která představuje počet prvních měsíců projektu. Stavový diagram postupu detekce je znázorněný na obrázku 4.5.

V detekci by také bylo vhodné rozlišit ty přispěvatele, kteří se na projektu podílí aktivně, například pravidelným přispíváním do programového kódu, a ty, kteří přispěli například pouze jedním komentářem. Do vyhodnocování přítomnosti anti-vzoru by pak byli zahrnuti pouze aktivní přispěvatelé. Tento přístup by vyžadoval hlubší definici toho, co je považováno za dostatečnou aktivitu. Do navrženého modelu detekce proto není zahrnutý.



Obrázek 4.5: Stavový diagram detekce Yet Another Programmer

5 Potřebná rozšíření aplikace

V souvislosti s rostoucím počtem detekovaných fenoménů v aplikaci SPAWn, bude provedeno její rozšíření o funkční vlastnosti. Rozšíření aplikace mají za cíl zvýšit udržitelnost a škálovatelnost aplikace při zmíněném nárůstu. Zároveň bude zvýšena konzistence informací v rámci frameworku SPADe.

V této kapitole budou navržena potřebná funkční rozšíření a změny ve struktuře aplikace. Následně bude popsána jejich implementace, změny ve struktuře projektu a v přidávání nových detekcí. Dále bude popsáno testování implementovaných částí a budou navrženy budoucí možná rozšíření.

5.1 Návrh rozšíření

V této podkapitole je popsán návrh funkčních rozšíření a jimi vyvolané změny ve struktuře aplikace.

5.1.1 Funkční rozšíření

Sada vhodných funkčních rozšíření byla vybrána se záměrem zlepšit konzistenci aplikace s dalšími částmi projektu SPADe, poskytnout uživateli důležité informaci o postupu detekce a také přidat možnosti znovupoužití zvolených konfigurací.

Dolování popisu z katalogu

Cílem tohoto rozšíření je zajistit automatickou těžbu popisu pro každý anti-vzor z online katalogu do aplikace. To má za cíl zajištění konzistence informací o anti-vzorech v aplikaci a v katalogu.

Ve výchozí podobě aplikace se popis anti-vzoru nachází na stránce detailu daného anti-vzoru, na který se uživatel dostane z hlavní stránky kliknutím na vybraný anti-vzor ze zobrazeného seznamu. V detailu anti-vzoru se nachází textové pole s popiskem „Anti-pattern Description“ obsahující stručný popis. Data do tohoto pole jsou vkládána z atributu instance třídy reprezentující anti-vzor.

Tímto funkčním rozšířením dojde k nahrazení popisu z atributu instance popisem z příslušné stránky v katalogu pro daný anti-vzor. Je nutné počítat s tím, že ne každý anti-vzor má svoji stránku v katalogu a z toho důvodu

bude z katalogu dolován popis pouze u těch, pro které stránka existuje. V opačném případě bude ponechán původní popis z atributu jejich instance.

Přihlašování uživatele

Pro následující rozšíření popsaná v dalších sekcích bude nutné implementovat přihlašování uživatele. To bude sloužit pro zpřístupnění některých sekcí a funkcí pouze přihlášeným uživatelům. Uskutečněno bude standardním přihlašovacím formulářem obsahujícím pole pro zadání jména a hesla. Po domluvě s vedoucím práce bude prozatím implementováno přihlašování jednoho uživatele (administrátora) a heslo nebude šifrováno. Přihlašování bude ovšem implementováno tak, aby bylo v budoucnu snadno rozšiřitelné o rozsáhlejší správu uživatelů, tedy přihlašování více uživatelů, kteří budou mít určené role, a tedy přidělená různá práva přístupu do různých částí aplikace.

Přidání popisu operacionalizace

Popis operacionalizace u každého anti-vzoru je důležitou informací pro uživatele. Dodáním této informace bude mít uživatel možnost zjistit, jak a na základě jakých dat detekce probíhala. Díky tomu může snadněji lokalizovat příčinu výskytu anti-vzoru, popř. posoudit, zda je operacionalizace adekvátní a přesná.

Jako vhodné místo pro umístění textového pole pro popis operacionalizace se nabízí stránka detailu jednotlivých anti-vzorů.

Pro vyčerpávající a přehledný popis operacionalizace nemusí stačit prosté textové pole s neformátovaným textem. Z tohoto důvodu bude pole umožňovat další funkce pro zvýšení přehlednosti. Konkrétně se jedná o formátování textu, tedy tučný text, kurzívu, podtrhávání, různou velikost fontu, formátování kódu, dále číslované a odrážkové seznamy, odkazy a v neposlední řadě obrázky pro poskytnutí různých grafů a diagramů.

Další otázkou je jak a kde daný popis operacionalizace vytvořit. Z toho důvodu bude do aplikace vložený editor, který umožní popis operacionalizace vytvářet nebo měnit přímo v aplikaci. Možnost upravovat pole s operacionalizací pomocí editoru bude pouze pro přihlášené uživatele do aplikace, a to z důvodu centralizace původu těchto úprav.

Ukládání konfigurací

Každý projekt je jedinečný a z toho důvodu jsou detekce anti-vzorů parametrizovatelné pomocí nastavování prahových hodnot. V původní podobě aplikace je možné prahové hodnoty pro detekce měnit, ale naráží se na určitá

úskalí. Prvním je, že nastavení prahových hodnot pro detekce je globální. To je problém v případě, že s aplikací pracuje více uživatelů zároveň, kdy jeden z nich může prahovou hodnotu změnit bez vědomí ostatních uživatelů, kteří nebudou poté dostávat jimi požadované výsledky. Dalším problémem je nemožnost uložení aktuálního nastavení pro pozdější použití na projektech podobného charakteru.

Řešením zmíněných problémů je implementace možnosti ukládání celých konfigurací. Konfigurace je v tomto smyslu chápána jako množina všech prahových hodnot pro všechny anti-vzory (tedy má jiný význam než entita Configuration v datovém skladu SPADe). Jedná se tedy o sadu prahových hodnot uložených pro opakované použití. Každá konfigurace bude mít svůj název a bude vybírána uživatelem. Různí uživatelé aplikace tak budou moci pracovat s různými konfiguracemi zároveň.

Výběr konfigurace bude probíhat uživatelem ze seznamu uložených konfigurací. Tento seznam by mohl být umístěný na stránce s nastavením jednotlivých prahových hodnot. Tímto způsobem by ale uživatel neměl možnost vidět, jakou konfiguraci má právě vybranou, pokud by se nacházel například na stránce s výběrem projektů a anti-vzorů k detekci nebo na stránce s výsledky detekce. Proto bude vhodné umístit výběr konfigurace do hlavičky každé stránky, tedy do lišty s navigací.

Přístup ke konfiguracím bude opět omezený pouze pro přihlášené uživatele. Ti budou moci konfigurace upravovat nebo vytvářet nové. Nepřihlášený uživatel bude mít k dispozici pouze jednu výchozí konfiguraci, která nebude v aplikaci měnitelná, a to ani přihlášenými uživateli. Výchozí konfigurace bude obsahovat množinu prahových hodnot použitelných pro velkou část projektů, ale nikoliv pro všechny.

5.1.2 Změny struktury aplikace

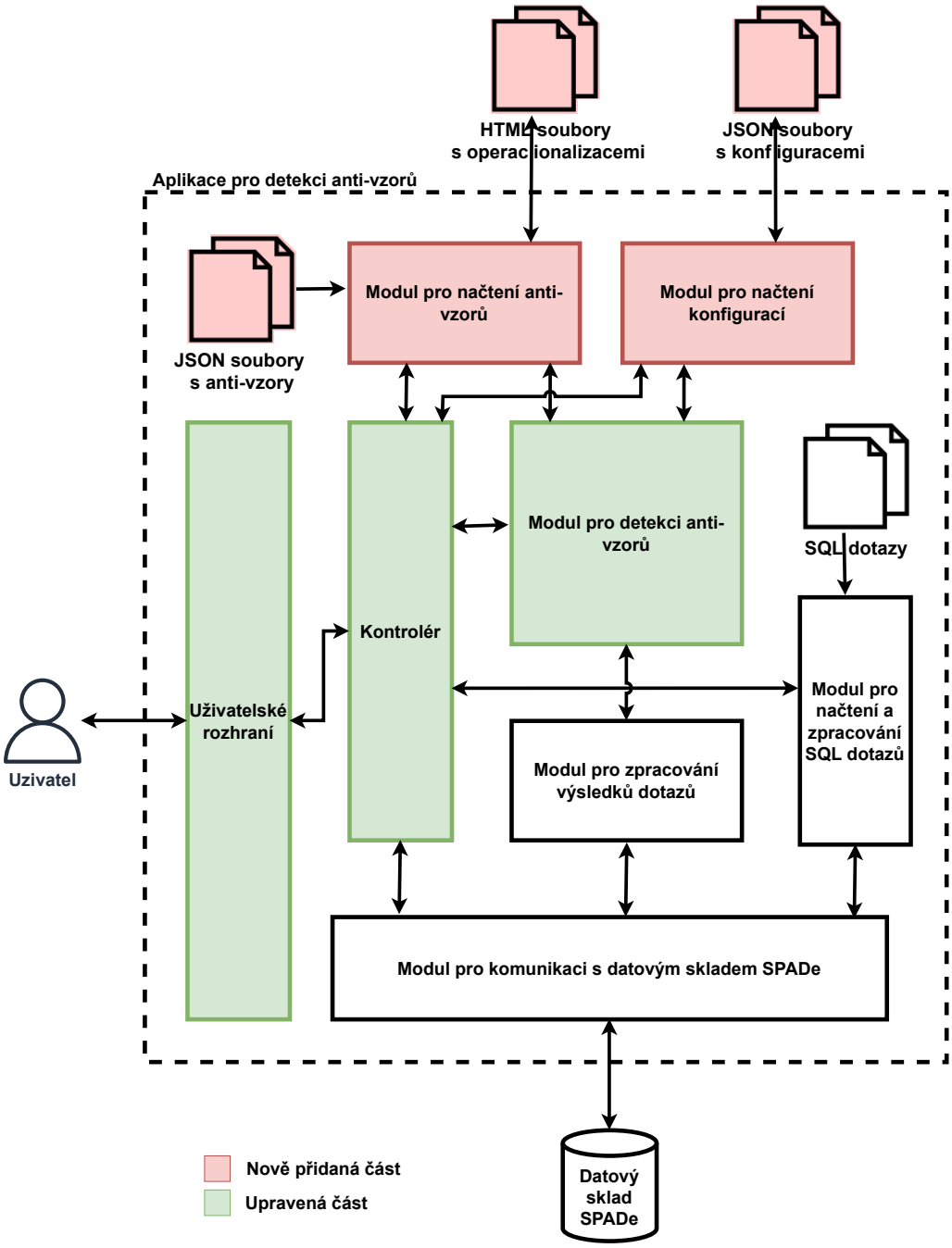
Spolu s funkčním rozšířením dojde také ke strukturálním změnám původní podoby aplikace. Některé změny ve struktuře jsou vyvolané stejnými faktory jako funkční změny a některé jsou následky implementace funkčních změn. Konkrétně se jedná o oddělení dat od implementace, tedy konkrétně oddělení informací o jednotlivých anti-vzorech do samostatných souborů, které budou součástí výsledné aplikace.

Zároveň dojde k vytvoření souborů pro ukládání konfigurací nesoucí sadu prahových hodnot, dále souborů pro ukládání operacionalizací a také složky pro obrázky v nich obsažených. Tyto soubory se budou nacházet v samostatných složkách mimo aplikaci.

Výsledná struktura aplikace je znázorněna na obrázku 5.1. Nově přidané

části jsou vyznačeny červenou barvou a původní části, které prošly úpravou, jsou zvýrazněny zelenou barvou. Aplikace bude rozšířena o tyto části:

- **Modul pro načítání anti-vzorů** – Modul zajišťuje načítání externích souborů s informacemi o anti-vzorech a souborů s popisem operacionalizace.
- **Soubory s anti-vzory** – Soubory obsahují informace o jednotlivých anti-vzorech, tedy jejich identifikátor, název, stručný popis a popis prahových hodnot pro jejich detekci.
- **Soubory s operacionalizacemi** – Soubory obsahují popis operacionalizace detekce anti-vzorů.
- **Modul pro práci s konfiguracemi** – Modul uchovává konfigurace ve vnitřní struktuře a zajišťuje čtení a zápis do externích souborů s konfiguracemi.
- **Soubory s konfiguracemi** – Soubory obsahují jednotlivé konfigurace, tedy jednotlivé sady prahových hodnot.



Obrázek 5.1: Výsledná struktura aplikace SPAWn

5.2 Implementace funkčních rozšíření

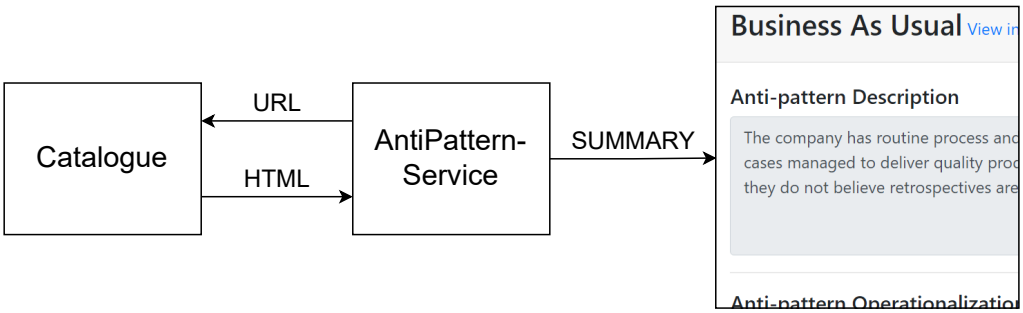
V následující části je popsána implementace navržených funkčních rozšíření do stávající aplikace SPAWn. Celkem bylo změněno nebo přidáno přibližně 1000 řádek kódu.

5.2.1 Dolování popisu z katalogu

Katalog obsahující popisy jednotlivých anti-vzorů existuje ve formě repozitáře na platformě GitHub[1]. Stránky s charakteristikou anti-vzorů jsou uloženy ve formátu Markdown (MD) souborů. Tyto soubory lze zobrazit v čistě textové formě, tedy zapsanou syntaxi pro MD soubor v neformátované podobě.

URL adresa MD souboru v textové podobě se skládá z pevné části, která je v aplikaci uložená v konstantě, a z proměnné části získané z atributu instance třídy představující anti-vzor. Tato URL adresa je předána metodě `getDescriptionFromCatalogue(...)` ve třídě `AntiPatternService`, která zajistí parsování HTML pomocí knihovních metod, a tím je získána kompletní charakteristika anti-vzoru. Pro parsování byla vybrána knihovna JSoup[2], která obsahuje veškeré potřebné metody. Alternativou by mohla být například knihovna HTMLCleaner.

Jako samotný popis anti-vzoru je z načteného souboru vybrána sekce „Summary“. Získaný text je prostřednictvím kontroléru zobrazený v uživatelském rozhraní v detailu anti-vzoru v poli „Anti-pattern Description“. Průběh dolování popisu je znázorněn na obrázku 5.2.



Obrázek 5.2: Průběh dolování popisu z katalogu

[1]https://github.com/ReliSA/Software-process-antipatterns-catalogue
[2]https://mvnrepository.com/artifact/org.jsoup/jsoup

5.2.2 Přihlašování uživatele

Pro přihlašování uživatele je implementován standardní přihlašovací formulář na samostatné stránce. Údaje z formuláře jsou ověřovány ve třídě `UserAccountServiceImpl`. Validní údaje pro přihlášení jsou uvedeny v souboru `application.properties`. Současná implementace je pouze dočasná a v budoucnu se počítá s jejím rozšířením o komplexnější správu uživatelů. Pro snadné rozšíření je v současné implementaci zavedeno rozhraní `UserAccountService`.

5.2.3 Přidání popisu operacionalizace

Pro popis operacionalizace bylo přidáno editovatelné pole do HTML stránky detailu anti-vzoru, konkrétněji je jedná o `<div contenteditable=true>`. Dále byl implementován editor pro úpravu tohoto pole. Editor je zobrazený formou lišty s nabídkou funkcí v horní části pole. Mezi implementované funkce patří:

- tučné písmo, kurzíva, podtržení,
- obalení tagem `code`,
- zarovnání vlevo, na střed, vpravo,
- nadpis stylu H1, H2, H3,
- číslovaný/odrážkový seznam,
- vložení hypertextového odkazu a obrázku.

Výše uvedené funkce byly implementovány za pomoci jazyka JavaScript v souboru `editor.js`.

Obalení tagem `code` je v editoru provedeno nahrazením vybrané části textu za stejný text obalený tímto tagem. K samotnému formátování pak dojde až po stisku tlačítka pro uložení popisu operacionalizace. Tento přístup byl zvolen z důvodu nejednotného chování různých prohlížečů při formátování `code` tagem.

Popis operacionalizace je rovněž možné formátovat přímým použitím HTML značek v poli s textem. Tím je možné dodat více pokročilé formátování nad rámec implementovaných funkcí.

Uložení

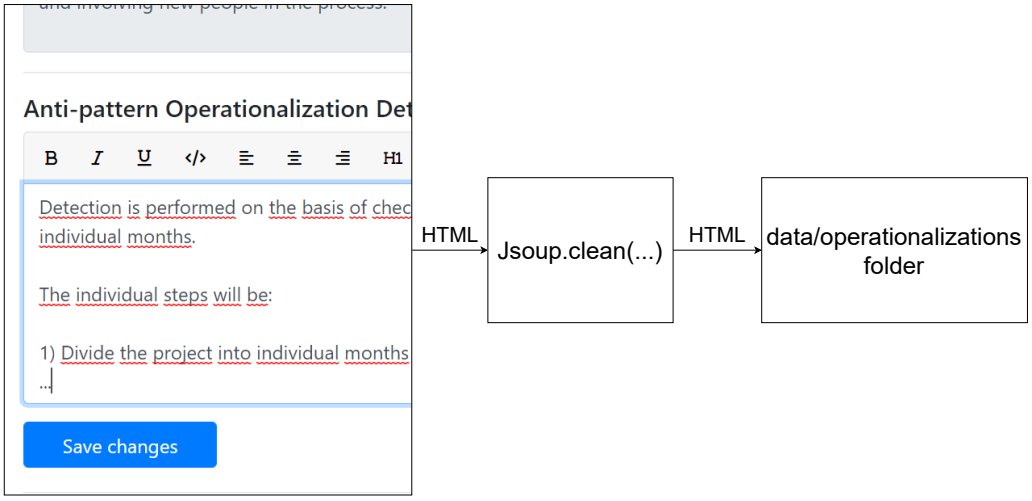
Pro uložení vytvořených popisů operacionalizací byl zvolen formát HTML, a to z důvodu jednoduchosti a upravitelnosti i mimo aplikaci. Při ukládání tak není nutné popis převádět do jiného formátu a stejný problém odpadá i u vkládání textu operacionalizace do připraveného pole na zobrazené stránce v prohlížeči. Výsledné soubory s popisem operacionalizace budou uloženy ve složce `data/operationalizations` a obrázky k nim pak ve složce `data/operationalizations/images`. Soubory s popisem je tak možné upravovat i mimo aplikaci, pokud k nim má uživatel přístup.

Průběh uložení popisu operacionalizace je znázorněn na obrázku 5.3

Bezpečnost

Ukládání popisů operacionalizací do souboru HTML a následné vkládání do obsahu souboru do pole při otevření stránky detailu anti-vzoru otevírá prostor pro takzvané Cross Site Scripting (XSS) útoky. Jedná se o typ útoku, při kterém dochází k vložení škodlivých skriptů do jinak důvěryhodných stránek. Škodlivý kód je pak součástí dynamického obsahu doručeného do prohlížeče uživatele, kde může dojít například ke zcizení citlivých údajů [21].

Pro odstranění případného škodlivého kódu byla využita knihovna JSoup a její metoda `Jsoup.clean(...)`. Tím dojde k odstranění potenciálně škodlivého kódu z popisu operacionalizace ještě před uložením do souboru. Odstraněny jsou například tagy `<script>` a jejich obsah nebo veškerá volání funkcí.



Obrázek 5.3: Průběh uložení popisu operacionalizace

5.2.4 Ukládání konfigurací

Pro ukládání konfigurací byl zvolen typ souboru JSON, který je oproti textovým strukturovaný a oproti XML snadněji čitelný. Pro čtení i zápis do JSON souborů byla využita knihovna Jackson[3]. Mezi možné alternativy patří například knihovny Gson nebo json-io.

Při spuštění aplikace jsou veškeré konfigurace z externích souborů přečteny a uloženy do vnitřní struktury. Vnitřní struktura pro ukládání konfigurací má tuto podobu:

```
Map<String, Map<String, Map<String, String>>>
                allConfigurations;
```

Nejvíc vnější mapa obsahuje název konfigurace a jako hodnota je mapa anti-vzorů. Klíčem v mapě anti-vzorů je název anti-vzoru a jako hodnota je mapa prahových hodnot. Poslední, nejvíc vnitřní mapa prahových hodnot je tvořena názvem prahové hodnoty v podobě klíče a hodnotou je samotná hodnota dané prahové hodnoty.

Veškeré metody pro práci s vnitřní strukturou s konfiguracemi se nacházejí ve třídě `ConfigurationRepository`, v balíku `repository`. V této třídě jsou také obsaženy metody pro čtení a zápis konfigurací do JSON souborů.

5.3 Změny způsobené rozšířením

V této podkapitole jsou podrobněji popsány pouze významné změny, které byly způsobeny výše popsaným rozšířením. Kromě toho byly v aplikaci provedeny i další, drobnější úpravy některých prvků uživatelského rozhraní. Jedná se například o stránku pro úpravu prahových hodnot v konfiguracích nebo o tabulku výběru množiny projektů a v nich detekovaných anti-vzorů.

5.3.1 Struktura projektu

Do struktury byla přidána složka `data` pro uložení konfigurací a operacionalizací anti-vzorů a složka `src/main/resources/antipatterns` pro popis anti-vzorů. Složka `queries` byla přesunuta. Rovněž byly přesunuty CSS a JavaScript soubory do samostatné složky `webapp/resources`.

V tabulce 5.1 je zobrazena výsledná struktura projektu aplikace SPAWn. Pro zvýšení přehlednosti a úspory místa v tabulce je znakem * nahrazena cesta `src/main` a znakem `#` je nahrazena složka `resources`.

[3]https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core

Složka/Soubor	*Obsah*
`data/configurations`	soubory s konfiguracemi
`data/operationalizations`	soubory s operacionalizacemi
`db`	skripty pro inicializaci datového skladu
`*/java`	zdrojové kódy aplikace
`*/#/application.properties`	nastavení aplikace
`*/#/queries`	definice dotazů pro anti-vzory
`*/#/antipatterns`	definice s popisem anti-vzorů
`*/webapp/WEB-INF/templates`	HTML šablony uživatelského rozhraní
`*/webapp/resources`	složka s CSS a JavaScript soubory
`pom.xml`	soubor pro sestavení aplikace
`Dockerfile`	soubor pro spuštění aplikace v Dockeru
`docker-compose.yml`	soubor pro definici služeb v Dockeru

Tabulka 5.1: Nová struktura projektu aplikace

5.3.2 Přidání dalších detekcí

Vlivem funkčních a strukturálních změn v aplikaci došlo ke změně při rozšiřování aplikace o detekce dalších anti-vzorů oproti návodu rozšíření z předchozí diplomové práce [26]. Pro úplnost je zde uvedený kompletní postup pro rozšíření o další detekce.

Pro implementaci rozšíření aplikace o další detekce je nutné provést následující kroky:

1. Vytvořit soubor ve formátu JSON ve složce `data/antipatterns` obsahující základní informace o anti-vzoru.

   Příklad struktury souboru je uvedený v příloze D.

2. Vytvořit třídu v balíku `detecting.detectors`, která implementuje rozhraní `AntiPatternDetector` ze stejného balíku.

   Rozhraní obsahuje čtyři metody, které je nutné implementovat. Jedná se o tyto metody:

   - `setAntiPattern(AntiPattern antiPattern)`,
   - `getAntiPatternModel()`,
   - `getSqlFileNames()`,
   - `getSqlQueries(List<String> queries)`,
   - `analzye(Project project, DatabaseConnection databaseConnection, Map<String, String> thresholds)`.

3. Vytvořit soubor nebo více souborů s SQL dotazy a umístit je do složky `src/resources/queries`.

   Pro správné fungování detekce je nutné zachovat jednotnou formu souborů s SQL dotazy. Každý dotaz musí být na samostatné řádce v samostatném souboru a musí být ukončen středníkem.

5.4 Testování

Testování funkčních rozšíření aplikace proběhlo formou jednotkových testů a uživatelského testování. Rozsah testování se omezuje jen na nově přidané části.

Jednotkové (JUnit) testy byly využity pro kontrolu správné funkčnosti metod pro práci s konfiguracemi a práci s JSON soubory. Celkem bylo napsáno 7 jednotkových testů, které pokrývají 26% nově implementovaného kódu a 9% z celkového kódu aplikace. Toto relativně nízké pokrytí jednotkovým testováním je způsobeno velkou mírou závislosti implementovaných rozšíření na externích zdrojích jako je databáze, online katalog a soubory s operacionalizacemi.

Uživatelské testování proběhlo formou testovacích scénářů a bylo provedeno čtyřmi externími testery. První dvojici testerů tvořili studenti informatiky, kteří byli předem seznámeni s implementací aplikace, zatímco druhá dvojice se skládala z osob bez technických znalostí, kteří s implementací seznámeni nebyli. Celkem byly provedeny čtyři uživatelské testy. Cílem uživatelských testů bylo ověřit správnou funkčnost přihlašování uživatele, správnost získaných dat při dolování z katalogu, správnost fungování všech funkcí v editoru operacionalizace a správnost ukládání prahových hodnot do původních a nových konfigurací. Testovací scénáře jsou přiloženy k této práci (umístění viz příloha A). Uživatelským testováním byla ověřena správná funkčnost implementovaných rozšíření a nebyly nalezeny žádné chyby.

5.5 Budoucí možná rozšíření

První z možných budoucích rozšíření aplikace SPAWn je rozsáhlejší správa uživatelů. V současné podobě existuje v aplikaci pouze jeden uživatelský účet a není možnost snadno přidat další. To by mohlo být rozšířeno o více uživatelských účtů, přidání jejich registrace přímo v aplikaci a nastavení uživatelských rolí s různými úrovněmi práv.

Další rozšíření by mohlo zahrnovat přidání hlubší charakteristiky anti-vzorů do aplikace, tedy i zobrazení příčin, následků a příbuzných anti-vzorů.

Tyto informace by pro zachování konzistence mohly být dolovány z online katalogu stejným způsobem, jako je tomu v případě popisů anti-vzorů.

Stránka s výsledky detekcí by mohla poskytovat detailnější informace, na základě kterých bylo rozhodnuto o přítomnosti daného anti-vzoru. V některých případech by také mohly být doplněny relevantní grafy. Zároveň by mohla být přidána možnost uložit získané výsledky pro pozdější použití, popřípadě možnost exportu výsledků do externího souboru (např. CSV nebo JSON) pro účely dalšího zpracování.

Významnému rozšíření aplikace by také přispěla implementace dalších detekcí anti-vzorů a špatných praktik.

6 Implementace detekce

V této kapitole je popsána implementace navržených modelů detekce vybraných anti-vzorů a špatných praktik z podkapitoly 4.2, popis prahových hodnot a jejich volba do výchozí konfigurace a popis testování správnosti implementovaných detekcí.

6.1 Implementace detekcí

V této podkapitole jsou popsány implementace navržených detekcí anti-vzorů a špatných praktik.

Unknown Poster

V prvním kroku detekce jsou vybrány veškeré osoby zapojené do zkoumaného projektu. Osoby jsou vybrány z pohledu `personView` podle hodnoty atributu `projectId`, která odpovídá identifikátoru pro vybraný projekt.

Dalším krokem je filtrace všech osob, které nemají uvedené celé jméno. To je provedeno kontrolou přítomnosti mezery v atributu `name`. V případě, že má každá osoba uvedené celé jméno, není anti-vzor detekován.

Pokud se v projektu nachází osoby bez celého jména, jsou pro tyto osoby prozkoumány jimi použité identity. Všechny identity vybrané osoby je možné získat z tabulky `identity`, kde se atribut `personId` rovná atributu `id` z pohledu `personView`. Pokud má osoba alespoň u jedné z identit vyplněno celé jméno v atributu `description` nebo vyplněný atribut `email`, je možné danou osobu identifikovat. Další možností je, že osoba nemá uvedeno celé jméno, ale má uvedenou přezdívku. Za identifikovatelné jsou považovány všechny přezdívky neobsahující podřetězce jako „unknown“ nebo „anonym“. Soubor těchto podřetězců je prahovou hodnotou anti-vzoru a je možné jej upravit nebo doplnit uživatelem.

V případě, že se v projektu nachází alespoň jedna osoba, kterou nelze na základě dostupných dat identifikovat, je anti-vzor prohlášen za detekovaný.

Bystander Apathy

Pro detekci je nejprve nutné získat všechny úkoly v daném projektu. Úkoly se nacházejí v tabulce `work_unit` a vybrány jsou na základě hodnoty atributu `projectId`.

Dalším krokem je získat počet přispěvatelů do daného úkolu. Jakákoli změna v projektu včetně identifikátoru autora změny se nachází v pohledu `configurationView`. Veškeré změny pro každý úkol se nacházejí v tabulce `work_item_change` a jsou vybrány na základě identifikátoru daného úkolu. V tabulce `configuraton_change` se pro tyto změny zjistí identifikátor konfigurace, do které patří. Z nalezených přispěvatelů je pro správnost detekce nutné vyfiltrovat automatické změny v projektu prováděné daným ALM nástrojem. Například v případě GitHubu se jedná o takzvaný Dependabot. Výčet jmen k filtraci je zadaný uživatelem jako prahová hodnota pro detekci.

Pokud má úkol pouze jednoho přispěvatele, na úkolu kromě jeho autora nikdo další nepracoval. V tomto případě byl detekován úkol s absencí spolupráce. Tvoří-li tyto úkoly významnou část z celkového počtu úkolů v projektu, je prohlášen anti-vzor Bystander Apathy za detekovaný. Hranice pro detekci je určena procentuálně a je zadaná uživatelem jako prahová hodnota.

Yet Another Programmer

Detekce je rozdělena na tři části. V první části jsou vybráni všichni přispěvatelé do projektu a je zjištěno datum jejich prvního zapojení. Datum prvního zapojení se zjistí z pohledu `configurationView`, kde je nejprve provedena filtrace na základě identifikátoru autora `authorId` a následně je vybráno datum prvního příspěvku podle atributu `created`.

V druhé části detekce je zjištěn začátek a konec projektu. Projekt je následně rozdělen na měsíce a ke každému měsíci jsou přiřazeny osoby, které se v tomto měsíci do projektu zapojily.

Poslední část detekce zahrnuje procházení měsíců a kontrolu počtu nově příchozích do projektu. Z toho jsou vynechány první měsíce projektu, jejichž konkrétní počet je určený uživatelem jako prahová hodnota. V případě, že alespoň v jednom měsíci se do projektu zapojilo větší množství členů, je anti-vzor detekovaný. Konkrétní počet členů pro detekci je určený uživatelem jako prahová hodnota.

6.2 Prahové hodnoty

V této podkapitole je popsán postup výběru prahových hodnot do výchozí konfigurace aplikace.

Unknown Poster

Nevalidní přezdívky – Mezi nevalidní přezdívky budou vybrány takové přezdívky, ze kterých není možné určit identitu osoby. Ve výchozí konfiguraci aplikace se jedná o přezdívky obsahující podřetězce „unknown“ a „anonym“.

Bystander Apathy

Minimální podíl úkolů s absencí spolupráce (`BA`) – Jakákoliv míra spolupráce by měla ideálně probíhat u většiny úkolů. Výjimkou mohou být málo náročné úkoly, například opravy nezávažných chyb. Do výchozí konfigurace byla zvolena hodnota 30%.

Nevalidní přispěvatelé – Mezi nevalidní přispěvatele budou patřit zejména automatické změny provedené používaným ALM nástrojem. Výchozí konfigurace bude zahrnovat řetězec „dependabot“, který je využívaný v nástroji GitHub.

Yet Another Programmer

Počet ignorovaných počátečních měsíců ($\texttt{YAP}_1$) – Počet počátečních měsíců projektu, ve kterých nebude detekce probíhat se bude vždy lišit dle rozsahu projektu. Do výchozí konfigurace byla zvolena hodnota 2.

Minimální měsíční nárůst pracovníků ($\texttt{YAP}_2$) – Výběr hodnoty této prahové proměnné bude záviset na rozsahu projektu a velikosti projektového týmu. Detekce budou probíhat zejména na open-source projektech na kterých se zpravidla podílí větší množství přispěvatelů, proto zvolená hodnota minimálního počtu nových členů pro detekci je 5.

6.3 Testování detekcí

Správná funkčnost implementace detekcí byla testována za pomoci nástroje Microsoft Excel. Test byl provedený kombinací ruční kontroly s využitím nástroje filtrace. Data ke kontrole byla získána dílčími SQL dotazy, které se však lišily od dotazů použitých v implementaci detekcí.

Každá z detekcí byla testována na datech dvou projektů. První z projektů obsahoval data z nástroje Redmine využitém ke studentským projektům v rámci předmětu Pokročilé softwarové inženýrství (KIV/ASWI). Druhý projekt obsahoval data z nástroje GitHub a jednalo se o open-source projekt.

Správnost detekcí byla potvrzena ve všech případech. Soubory s průběhem a výsledky jednotlivých testů jsou přiloženy k práci (umístění viz příloha A).

7 Experiment

V této kapitole bude proveden experiment detekce na datech z vybraných open-source projektů. Nejprve budou vybrány vhodné projekty a nastaveny prahové hodnoty pro detekce. Následně bude spuštěna detekce v aplikaci SPAWn nad množinou projektových dat v datovém skladu SPADe. Poté budou popsány výsledky detekce a bude provedeno zhodnocení dosažených výsledků.

7.1 Výběr projektů

Experiment bude probíhat nad daty z open-source projektů. Aby byla detekce proveditelná a efektivní, byla pro výběr projektů určena tato kritéria:

1. projekt se nachází v ALM nástroji, pro který je ve frameworku SPADe nadefinována datová pumpa,
2. do projektu je zapojeno více členů než jeden,
3. data jsou dostupná pro správu verzí i změn.

Jako vhodný ALM nástroj, ze kterého budou projekty vybírány, byl zvolen GitHub, a to z důvodu existence datové pumpy pro Git úložiště (které GitHub projekty používají) i pro GitHub úkoly. Projekty z nástroje byly vybírány náhodně z různých tematických oblastí tak, aby počty přispěvatelů a úkolů v nich byly dostatečné, ale zároveň nebyly příliš vysoké, a získané výsledky detekcí tak bylo možné ručně ověřit, popřípadě aby těžení dat netrvalo příliš dlouho. Celkem bylo vybráno šest open-source projektů. Vybrané projekty a k nim zvolený identifikátor používaný v tomto experimentu se nachází v tabulce 7.1.

V prvním kroku byla projektová data z nástroje vytěžena do datového skladu SPADe (tento krok procesu nebyl součástí práce). Těžení dat proběhlo 29. 4. 2022. V tabulce 7.2 se nachází základní parametry vybraných projektů. Z důvodu úspory místa v tabulce je znakem * nahrazen prefix všech URL projektů, *https://github.com/*.

#1	**Název:** arborist **URL:** */npm/arborist
#2	**Název:** Barcode Scanner **URL:** */hyperoslo/BarcodeScanner
#3	**Název:** CipherSweet **URL:** */paragonie/ciphersweet
#4	**Název:** Corona Tracker **URL:** */mhdhejazi/CoronaTracker
#5	**Název:** Google Maps SDK for iOS Utility Library **URL:** */googlemaps/google-maps-ios-utils
#6	**Název:** Responsive HTML email signature(s) **URL:** */danmindru/responsive-html-email-signature

Tabulka 7.1: Vybrané projekty

#	Datum začátku projektu	Počet přispěvatelů	Počet úkolů
1	13.08.2014	246	134
2	18.03.2016	203	129
3	23.04.2018	33	36
4	02.03.2020	111	74
5	30.08.2013	531	267
6	12.01.2015	52	71

Tabulka 7.2: Parametry vybraných projektů

7.2 Nastavení prahových hodnot

Unknown Poster

Nástroj GitHub umožňuje přispívat pouze přihlášeným uživatelům a v datech by se tedy neměly nacházet neidentifikovatelné identity. Z toho důvodu bude použita výchozí konfigurace.

Bystander Apathy

Nastavení nevalidních přispěvatelů z výchozí konfigurace obsahuje řetězec „dependabot“. Kromě toho platforma GitHub obsahuje další automatické změny prováděné pod identitou „GitHub“. Konfigurace nevalidních přispěvatelů bude tedy obsahovat dvojici řetězců „dependabot“ a „GitHub“.

Míra úkolů bez spolupráce je zadána relativně a není závislá na celkovém počtu úkolů, tedy není nutné měnit hodnotu `BA` z výchozí konfigurace pro jednotlivé projekty.

Yet Another Programmer

Pro vhodné určení prahové hodnoty $\texttt{YAP}_1$ vylučující z detekce první fázi projektu by bylo nutné dobře znát kontext a průběh celého projektu, tedy přesně identifikovat tuto první fázi. To z dostupných dat není možné, a proto bude použita hodnota z výchozí konfigurace. Aby výsledky nebyly zkreslené, bylo by možné tuto hodnotu snížit tak, aby nepokrývala příliš velkou část z trvání projektu. Veškeré projekty jsou však dostatečně dlouhé, z toho důvodu není nutné výchozí hodnotu upravovat.

Prahová hodnota pro minimální počet nových pracovníků $\texttt{YAP}_2$ je zadána absolutním číslem. Ve zvolených projektech jsou v počtech pracovníků velké rozdíly, a proto bude prahová hodnota nastavena na základě těchto počtů. Projekty budou rozděleny do skupin, viz tabulka 7.3.

Skupina	Počet přispěvatelů	Prahová hodnota $\texttt{YAP}_2$
`A`	<0;100)	5
`B`	<100;200)	8
`C`	<200;600)	10

Tabulka 7.3: Nastavení prahové hodnoty $\texttt{YAP}_2$

7.3 Výsledky

V tabulce 7.4 jsou zobrazeny výsledky nově implementovaných detekcí pro jednotlivé projekty.

#	Unknown Poster	Bystander Apathy	Yet Another Programmer
1	✗	✓	✗
2	✗	✗	✓
3	✗	✗	✗
4	✗	✗	✓
5	✗	✗	✓
6	✗	✓	✗

✓ – detekováno, ✗ – nedetekováno

Tabulka 7.4: Výsledky experimentu

Unknown Poster

Detekce potvrdila předpokládaný výsledek. V žádném z projektů nebyl anti-vzor Unknown Poster detekován.

Bystander Apathy

Výskyt anti-vzoru Bystander Apathy byl potvrzen u dvou projektů. Úkoly s absenci spolupráce v těchto projektech dosahovaly poměrů 50% a 43,66% z celkového počtu úkolů. Výsledky pro jednotlivé projekty, včetně prahové hodnoty `BA` pro detekci jsou zobrazeny v grafu 7.1.

Yet Another Programmer

Výskyt anti-vzoru Yet Another Programmer byl potvrzený u tří z šesti zkoumaných projektů. Ve dvou z těchto projektů byl detekován výrazný nárůst přispěvatelů pouze v jednom měsíci. Ve třetím projektu byl výskyt detekován v 11 z celkových 104 měsíců. Přírůstky nových přispěvatelů v jednotlivých projektech rozdělených do skupin, včetně příslušných prahových hodnot $\texttt{YAP}_2$, jsou zobrazeny v grafech na obrázku 7.2.

7.4 Zhodnocení

Unknown Poster

Anti-vzor Unknown Poster nebyl detekovaný v žádném ze zkoumaných projektů. Tento výsledek vychází z vlastnosti nástroje GitHub, kde je pro jakýkoliv příspěvek do kteréhokoli projektu nutno mít v nástroji vytvořený účet a být přihlášený. Ke každému příspěvku lze tedy jednoznačně určit jeho autora v podobě příslušného uživatele.

Je však nutné zmínit, že i přes použití uživatelských účtů vystupuje každá osoba do určité míry anonymně. Použité uživatelské jméno a email nemusí nutně vypovídat o reálné identitě osoby. Do open-source projektu může přispívat kterákoli osoba pod různými uživatelskými účty. Určit tak reálnou identitu všech přispěvatelů je v případě open-source projektu takřka nemožné. U komerčních projektů je naopak přístup do projektu pouze pro vybrané uživatele, tedy přístup do projektu je umožněn těm uživatelským účtům, pro které jsou zpravidla známi jejich vlastníci.

Bystander Apathy

Po ruční kontrole bylo zjištěno, že úkoly detekované jakožto úkoly bez probíhající spolupráce se týkaly zejména menších změn v projektu, často se jednalo o opravy chyb nebo úpravy detailů. Úkoly ve kterých byly diskutovány klíčové prvky projektu a implementovány funkční vlastnosti zpravidla obsahovaly spolupráci více členů.

Výsledek detekce na vybrané množině projektů lze tedy interpretovat tak, že spolupráce na každém projektu probíhala ve velké míře a detekce anti-vzoru ve dvou případech byla způsobena především vysokým poměrem málo rozsáhlých úkolů, které spolupráci často nevyžadovaly.

I přesto by však bylo vhodné, aby i méně rozsáhlé úkoly obsahovaly účast dalšího přispěvatele, který by například ověřil řešení a jeho správnost potvrdil v komentáři. Tímto přístupem by došlo k validaci řešení úkolu a vyšší pravděpodobnosti odhalení případné chyby.

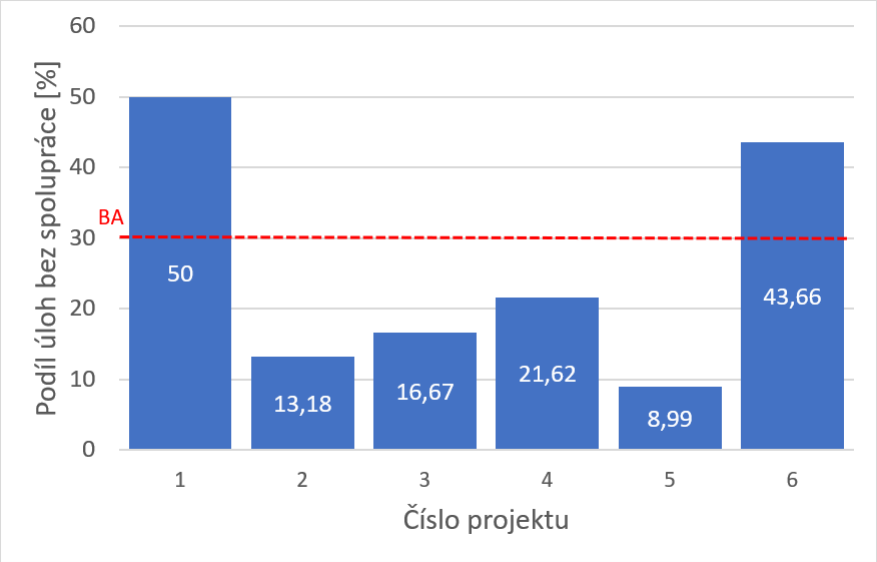
Yet Another Programmer

Výsledek detekce je ovlivněný zejména samotnou povahou open-source projektů, které často nejsou striktně omezeny časem ani rozsahem, tedy práce může probíhat i poté, co je dosaženo hlavního záměru. Další práce na projektu často zahrnují opravy nalezených chyb a různá vylepšení stávajícího řešení (tedy dlouhodobá podpora, údržba a obecný provoz). Zároveň u těchto

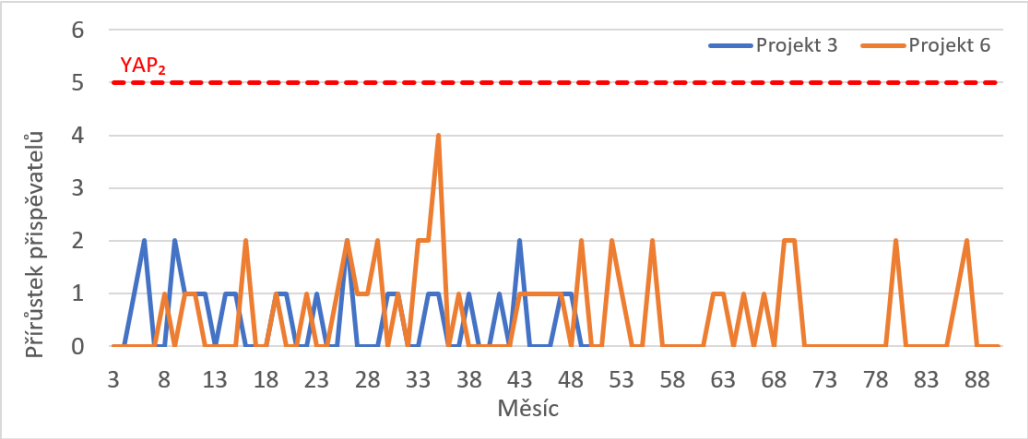
projektů není přímo určený projektový tým a do projektu se může zapojit kdokoliv v jakékoli fázi, a to třeba jen v podobě komentáře.

Pro vyhodnocení, zda má detekovaný anti-vzor skutečně negativní dopad na další průběh a dokončení projektu, je zapotřebí dobře znát kontext pro daný open-source projekt. V případě, že v projektu nejsou nastaveny žádné limity pro rozsah a datum dokončení, nemusí tak mít výskyt tohoto anti-vzoru žádný negativní vliv a může se dokonce jednat o pozitivum ve formě zvýšeného zájmu potenciálních uživatelů ať už produktu nebo báze kódu (možný případ projektů 1 a 5).

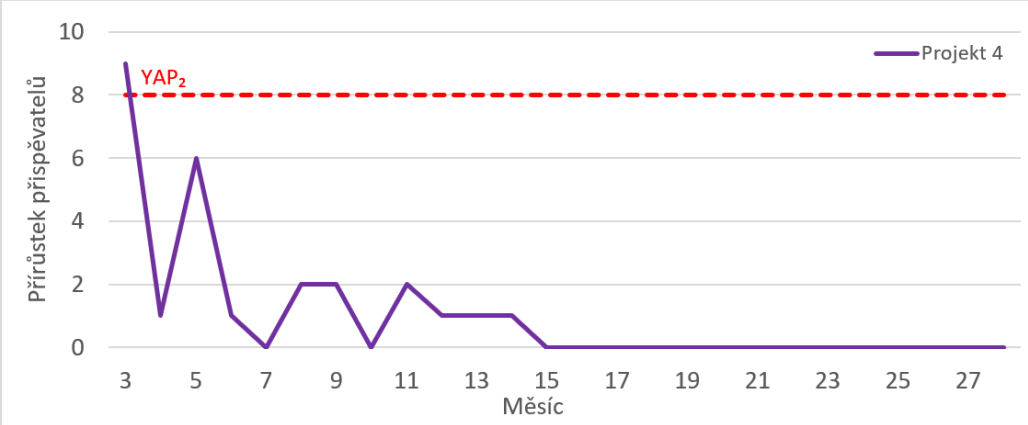
U projektu 4 byl detekovaný výrazný nárůst nových přispěvatelů hned v prvním zkoumaném (celkově třetím) měsíci. V tomto případě se zdá, že prahová hodnota ignorování prvních dvou měsíců nebyla ideální, tedy nebyla do ní zahrnuta celá počáteční fáze projektu. Bylo by proto vhodné blíže prozkoumat kontext této části projektu a případně detekci opakovat s novou prahovou hodnotou.



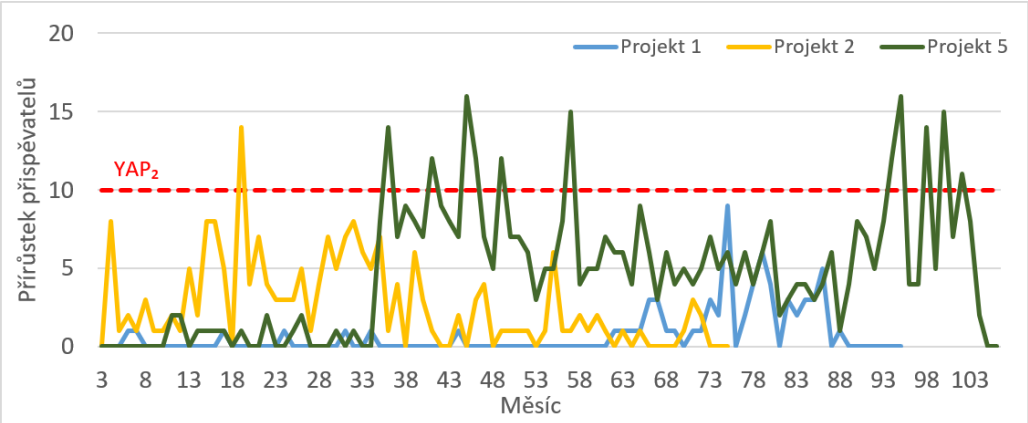
Obrázek 7.1: Poměr úloh bez spolupráce v projektech



(a) Skupina A



(b) Skupina B



(c) Skupina C

Obrázek 7.2: Přírůstky přispěvatelů v jednotlivých měsících

8 Závěr

Cílem této bakalářské práce bylo navržení modelů špatných praktik projektového řízení v open-source projektech a implementace jejich automatické detekce do frameworku Software Anti-pattern Detector (SPADe), konkrétně do webové aplikace SPADe Web Interface (SPAWn).

Za účelem splnění cíle práce bylo nutné seznámit se s problematikou špatných praktik a anti-vzorů v projektovém řízení softwarových projektů (kapitola 2). Následně byl prozkoumán framework SPADe (podkapitola 3.3). V další části byly vybrány vhodné špatné praktiky k detekci na základě dostupných projektových dat a zároveň byly navrženy modely pro jejich zachycení (podkapitola 4.2). Navržené modely, včetně potřebných funkčních rozšíření, byly implementovány do aplikace SPAWn (kapitoly 5 a 6). V poslední části byl provedený experiment detekce na množině vybraných open-source projektů následovaný zhodnocením a diskuzí nad získanými výsledky (kapitola 7).

Výstupy této práce představují rozšíření aplikace SPAWn o sadu dalších automatických detekcí. Operacionalizované modely vybraných špatných praktik ne vždy zachycují anti-vzory v největší možné míře, ale i tak jejich detekce vede na zajímavé poznatky. Příkladem může být odhalení výrazného nárůstu nových pracovníků, které nastalo v různých fázích projektů a z různých příčin. Implementace navržených modelů je ověřená, protože k jejímu testování bylo použito nezávislých SQL dotazů, které se lišily od těch v detekci použitých, v kombinaci s nástrojem pro filtraci dat a ruční kontrolou. Kromě toho byly do aplikace implementovány potřebné funkční prvky, které mají za cíl zvýšit použitelnost, udržitelnost a škálovatelnost aplikace.

Provedený experiment vedl na přínosné poznatky a úvahy a při jeho rozšíření (testování na více projektech, detailnější charakteristiky a operacionalizace anti-vzorů atd.) může mít významný výzkumný potenciál.

Zadání práce bylo splněno v plném rozsahu.

Literatura

[1] Alexander, M. *What is a project manager? The lead role for project success* [online]. CIO, 2021. [cit. 2022/04/20]. Dostupné z: https://www.cio.com/article/230682/what-is-a-project-manager-the-lead-role-for-project-success.html.

[2] Ambler, S. – Holitza, M. *Agile For Dummies, IBM Limited Edition.* John Wiley & Sons, Inc, 2012. ISBN 978-1-118-30506-5.

[3] Ambler, S. W. *Common Role Anti-Patterns in Online Discussion Forums* [online]. 2022. [cit. 2022/06/09]. Dostupné z: http://www.ambysoft.com/essays/discussionListAntiPatterns.html.

[4] Bezděk, P. *Identifikace a modelování špatných praktik v projektovém řízení. Diplomová práce.* Katedra informatiky a výpočetní techniky, Západočeská univerzita v Plzni, 2019. Dostupné z: http://hdl.handle.net/11025/39185.

[5] Bhandari, P. *Operationalization | A Guide with Examples, Pros and Cons* [online]. Scribbr, 2020. [cit. 2022/04/23]. Dostupné z: https://www.scribbr.com/dissertation/operationalization/.

[6] Brown, W. et al. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis.* John Wiley & Sons, Inc., 1998. ISBN 0-471-19713-0.

[7] Cohen, E. *The Definitive Guide to Project Management Methodologies* [online]. Workamajig, 2019. [cit. 2022/04/20]. Dostupné z: https://www.workamajig.com/blog/project-management-methodologies.

[8] Coplien, J. O. – Harrison, N. B. *Organizational Patterns of Agile Software Development.* Prentice-Hall, Inc., 2004. ISBN 978-0-13-146740-8.

[9] Danel, R. *Datový sklad.* Vysoká škola báňská – Technická univerzita Ostrava, 2010. Dostupné z: https://homel.vsb.cz/~dan11/is_skripta/IS%202010%20-%20Danel%20-%20Datovy%20sklad.pdf.

[10] Easy, T. *Co je agilní vývoj a proč ho používáme?* [online]. Think Easy, 2020. [cit. 2022/05/07]. Dostupné z: https://thinkeasy.cz/co-je-agilni-vyvoj-a-proc-ho-pouzivame/.

[11] Education, I. C. *What is ETL?* [online]. IBM, 2020. [cit. 2022/04/22]. Dostupné z: https://www.ibm.com/cloud/learn/etl.

[12] Ilieş, E. – Crişan, E. – Mureşan, I. N. Best Practices in Project Management. *Review of International Comparative Management*. March 2010, 11, s. 9. Dostupné z: http://www.rmci.ase.ro/no11vol1/Vol11_No1_Article4.pdf.

[13] Institute, P. M. *A Guide to the Project Management Body of Knowledge (4th Ed.)*. Project Management Institute, Inc., 2008. ISBN 978-1-933890-51-7.

[14] Jenkins, N. *A Project Management Primer: Basic Principles - Scope Triangle* [online]. ProjectSmart. [cit. 2022/04/20]. Dostupné z: https://www.projectsmart.co.uk/best-practice/project-management-scope-triangle.php.

[15] Kroll, P. – Kruchten, P. *Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0-321-16609-4.

[16] ManagementMania. *Řízení projektů (Project Management)* [online]. ManagementMania.com, 2016. [cit. 2022/04/20]. Dostupné z: https://managementmania.com/cs/metody-rizeni-projektu.

[17] Pecinovský, R. *Návrhové vzory*. Computer Press, 2015. ISBN 978-80-251-1582-4.

[18] Projects, L. F. *Home – CHAOSS* [online]. The Linux Foundation, 2022. [cit. 2022/05/27]. Dostupné z: https://chaoss.community/.

[19] Pícha, P. *Datový model nástroje SPADe a jeho mapování na projektová data z ALM nástrojů. Práce ke zkoušce z předmětu Moderní programovací styly a metody.* Katedra informatiky a výpočetní techniky, Západočeská univerzita v Plzni, 2016.

[20] Pícha, P. *Detecting software development process patterns in project data. Technická zpráva DCSE/TR-2019-04*. Katedra informatiky a výpočetní techniky, Západočeská univerzita v Plzni, 2019. Dostupné z: https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2019/Rigo_P%c3%adcha_2019_4.pdf.

[21] Rosencrance, L. *What is cross-site scripting (XSS)?* [online]. OWASP, 2021. [cit. 2022/04/23]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/cross-site-scripting.

[22] Sahoo, J. *What is a CI/CD Pipeline ? Continuous Integration and Continuous Delivery Explained.* [online]. OpsMx, 2021. [cit. 2022/04/21]. Dostupné z: https://www.opsmx.com/blog/what-is-a-ci-cd-pipeline/.

[23] Smartsheet. *Waterfall* [online]. Smartsheet, 2022. [cit. 2022/05/27]. Dostupné z: https://www.smartsheet.com/content-center/best-practices/project-management/project-management-guide/waterfall-methodology.

[24] Tamburri, D. A. – Palomba, F. – A., Z. Discovering Community Patterns in Open-Source: A Systematic Approach and Its Evaluation. *Empirical Software Engineering*. 2018.

[25] Tran, L. *Worst Practice in Project Management* [online]. InLoox, 2022. [cit. 2022/04/20]. Dostupné z: https://www.inloox.com/company/blog/articles/worst-practice-in-project-management.

[26] Váně, O. *Analýza přítomnosti anti-vzorů v datech nástrojů pro řízení projektů. Diplomová práce*. Katedra informatiky a výpočetní techniky, Západočeská univerzita v Plzni, 2021. Dostupné z: http://hdl.handle.net/11025/44774.

[27] Watts, A. *Project Management (2nd Ed.)*. BCcampus, 2014. ISBN 978-1-77420-013-1.

[28] Wikipedia. *Bystander effect* [online]. 2022. [cit. 2022/06/09]. Dostupné z: https://en.wikipedia.org/wiki/Bystander_effect.

[29] Xexeo, G. *Yet Another Programmer* [online]. 2010. [cit. 2022/06/09]. Dostupné z: https://wiki.c2.com/?YetAnotherProgrammer.

Přehled zkratek

- **ALM** – Application Lifecycle Management
- **ASWI** – Pokročilé softwarové inženýrství
- **CD** – Continuous Deployment
- **CI** – Continuous Integration
- **CSS** – Cascading Style Sheets
- **CSV** – Comma-separated Values
- **ETL** – Extract-Transform-Load
- **FAV** – Fakulta aplikovaných věd
- **HTML** – Hypertext Markup Language
- **JDBC** – Java Database Connectivity
- **JSON** – JavaScript Object Notation
- **KIV** – Katedra informatiky a výpočetní techniky
- **MD** – MarkDown
- **RUP** – Rational Unified Process
- **SPADe** – Software Process Anti-pattern Detector
- **SPAWn** – SPADe Web Interface
- **SQL** – Structured Query Language
- **UP** – Unified Process
- **URL** – Uniform Resource Locator
- **VCS** – Version Control System
- **XML** – Extensible Markup Language
- **XSS** – Cross Site Scripting
- **ZČU** – Západočeská univerzita v Plzni

Seznam obrázků

Seznam tabulek

A Obsah ZIP souboru

V následujícím seznamu je popsána struktura a obsah ZIP souboru.

- **Text_prace**
  - **zdroj** – zdrojové kódy textu a použité obrázky
  - **bakalarska_prace.pdf** – text bakalářské práce
- **Aplikace**
  - **AntiPatternDetectionApp**
    - **data**
      - **configurations** – konfigurace prahových hodnot
      - **operationalizations** – popisy operacionalizací anti-vzorů
    - **db**
      - **spade.sql** – struktura databáze
      - **spade-views.sql** – databázové pohledy
      - **spade-config.sql** – konfigurace databáze
    - **java_doc** – dokumentace aplikace
    - **src** – zdrojové kódy aplikace
    - **Dockerfile** – příkazy k sestavení image v Dockeru
    - **docker-compose.yml** – konfigurace pro Docker Compose
    - **pom.xml** – konfigurace pro Maven
    - **uploads.ini** – konfigurace pro nahrávání souborů
- **Vstupni_data**
  - **db**
    - **spade-data.sql** – data projektů
- **Vysledky**
  - **uzivatelske_testy** – výsledky uživatelských testů
  - **testy_detekci** – výsledky testů detekcí
- **Readme.txt** – popis obsahu a struktury ZIP souboru

B Instalační příručka

V této příručce je popsán postup instalace aplikace SPAWn na operačním systému Windows.

B.1 Potřebné nástroje

Pro spuštění aplikace jsou zapotřebí tyto nástroje:

- Docker[1],
- Docker Compose[2].

B.2 Postup instalace a spuštení

Spuštění aplikace se provede následujícím postupem:

1. Zkopírovat složku `AntiPatternDetectionApp` z přiloženého ZIP souboru na libovolné místo.
2. Zkopírovat soubor `Vstupni_data/db/spade-data.sql` z přiloženého ZIP souboru do složky `AntiPatternDetectionApp/db` na zvoleném místě z přechozího bodu.
3. Otevřít příkazový řádek a přesunout se do kořenové složky projektu s názvem `AntiPatternDetectionApp` na zvoleném místě.
4. Provést konfiguraci aplikace podle podkapitoly B.3.
5. Vytvořit potřebné komponenty aplikace následujícím příkazem.

```
docker-compose build
```

6. Spustit aplikaci následujícím příkazem.

```
docker-compose up
```

7. Provést obnovu a konfiguraci databáze podle podkapitoly B.4.

[1] https://docs.docker.com/get-docker
[2] https://docs.docker.com/compose/install

B.3 Konfigurace aplikace

Konfiguraci aplikace je možné provést v souborech `docker-compose.yml` a `application.properties`.

docker-compose.yml

V tomto souboru je možné nakonfigurovat porty, na kterých budou spuštěny jednotlivé služby. Dále je v tomto souboru možné nastavit jméno a heslo pro připojení k databázi a k jejímu webovému rozhraní. Soubor se nachází v kořenovém adresáři projektu a obsah jeho aktuální verze je zobrazen níže.

```
version: '3.3'

services:
  #service 1: definition of mysql database
  db:
    image: mysql:latest
    container_name: mysql-db
    environment:
      - MYSQL_ROOT_PASSWORD=<password for root>
    ports:
      - "3306:3306"
    restart: always
    volumes:
      - ./db/spade.sql:/usr/local/etc/spade.sql
      - ./db/spade-views.sql:/usr/local/etc/spade-views.sql
      - ./db/spade-config.sql:/usr/local/etc/spade-config.sql
      - ./db/spade-data.sql:/usr/local/etc/spade-data.sql

  #service 2: definition of phpMyAdmin
  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    container_name: my-php-myadmin
    ports:
      - "8082:80"
    restart: always
    depends_on:
      - db
    environment:
      SPRING_DATASOURCE_USERNAME: <username for phpMyAdmin>
      SPRING_DATASOURCE_PASSWORD: <password for phpMyAdmin>
    volumes:
      - ./uploads.ini:/usr/local/etc/php/conf.d/uploads.ini
    # add configuration due to max upload file size


```

```
  #service 3: definition of your spring-boot app
  antipatterndetection:
    image: anti-pattern-detection
    container_name: anti-pattern-detection-app
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8080:8080"                          # spring app will
   starts on port 8080 (<output_port:port_inside_of_docker>)
    restart: always

    depends_on:
      - db
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql-db:3306/spade
      SPRING_DATASOURCE_USERNAME: <username for DB connection
   >
      SPRING_DATASOURCE_PASSWORD: <password for DB connection
   >
      DATA_PATH: usr/local/etc/data/
    volumes:
      - ./data/:/usr/local/etc/data/
      - ./src/main/resources/application.properties:/
   application.properties
```

application.properties

V tomto souboru je možné nastavit údaje pro připojení k MySQL databázi a přihlašovací údaje administrátora aplikace. Soubor se nachází ve složce `./src/main/resources` a jeho obsah je zobrazen níže.

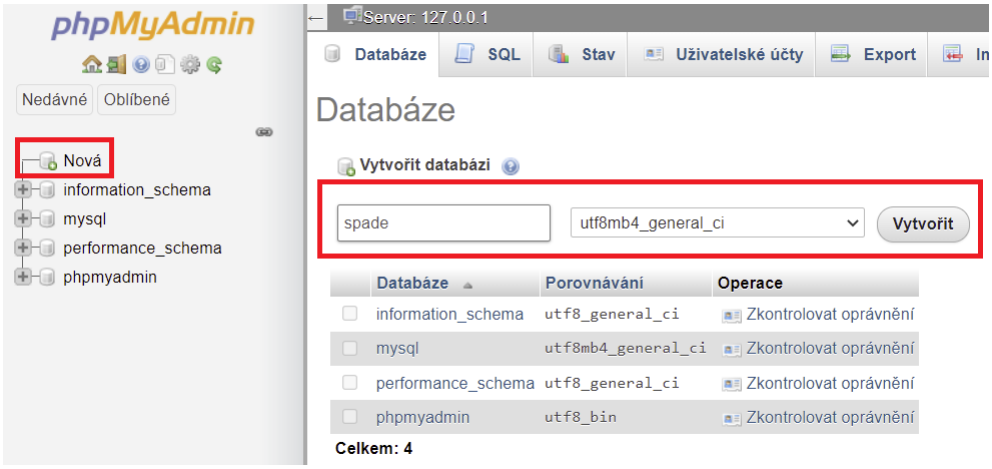
```
# database connection URL address:
spring.datasource.url=jdbc:mysql://localhost:3306/spade

# database connection username:
spring.datasource.username=

# database connection password:
spring.datasource.password=

# admin account username:
account.user.name=

# admin account password:
account.user.password=
```

B.4 Obnova a konfigurace databáze

Tato sekce popisuje dva možné způsoby pro obnovení databáze ze zálohy a její následnou konfiguraci.

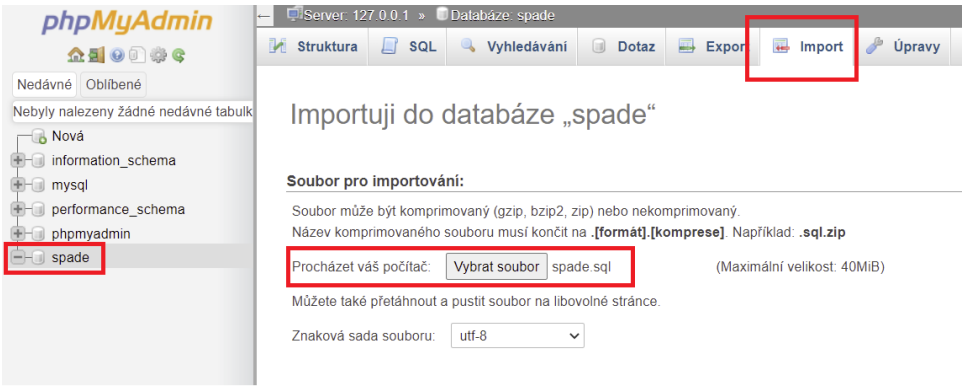
Pomocí rozhraní phpMyAdmin

1. Otevřít rozhraní phpMyAdmin, které běží na portu `8082`.
2. Přihlásit se do rozhraní pomocí údajů nastavených při konfiguraci aplikace (viz podkapitola B.3).
3. Vytvořit novou databázi. Tato databáze bude mít název `spade` a kódování `utf8mb4_general_ci`. Postup je znázorněný na obrázku B.1.



Obrázek B.1: Vytvoření databáze

4. Zvolit databázi `spade` z výběru všech databází. Přejít do sekce *Import*, stisknout tlačítko *Vybrat soubor* a vybrat soubor `spade.sql` ze složky `db` v kořenovém adresáři projektu. Následně stisknout tlačítko *Proveď* a vyčkat na zpracování souboru. To může trvat i několik minut. Postup je znázorněn na obrázku B.2. Tímto krokem se vytvoří struktura databáze.



Obrázek B.2: Obnova databáze ze zálohy

5. Provést krok č. 4 pro soubory `spade-views.sql`, `spade-config.sql` a `spade-data.sql`. Všechny soubory se nacházejí ve složce `db` v kořenovém adresáři projektu. Tím se vytvoří potřebné pohledy, databáze bude nakonfigurovaná, naplněna daty a aplikace bude plně funkční.

Pomocí příkazového řádku

1. V příkazovém řádku spustit následující příkaz:

```
docker exec -it mysql-db bin/bash
```

2. Příhlásit se do databáze pomocí následujícího příkazu:

```
mysql -u <uzivatel> -p
```

3. Zadat heslo pro uživatele `root`:

Heslo: `<heslo>`

4. Vytvořit novou databázi příkazem:

```
create database spade;
```

5. Přepnout se do nově vytvořné databáze příkazem:

```
use spade;
```

6. Vytvořit strukturu databáze příkazem:

```
source /usr/local/etc/spade.sql
```

7. Vytvořit potřebné pohledy příkazem:

```
source /usr/local/etc/spade-views.sql
```

8. Nakonfigurovat databázi příkazem:

```
source /usr/local/etc/spade-config.sql
```

9. Naplnit databázi daty příkazem:

```
source /usr/local/etc/spade-data.sql
```

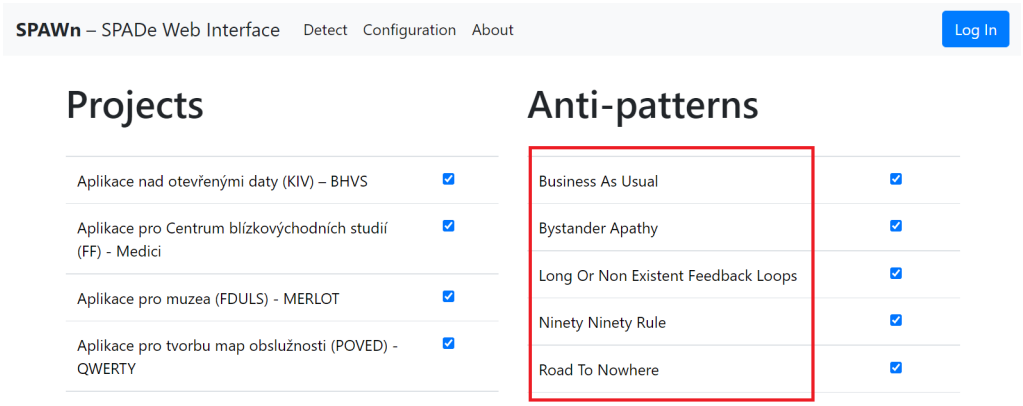
Nyní je databáze nakonfigurovaná a aplikace plně funkční.

C Uživatelská příručka

V této příručce je popsáno ovládání nově přidaných prvků aplikace. Ovládání původních funkcí je možné najít v předešlé diplomové práci [26].

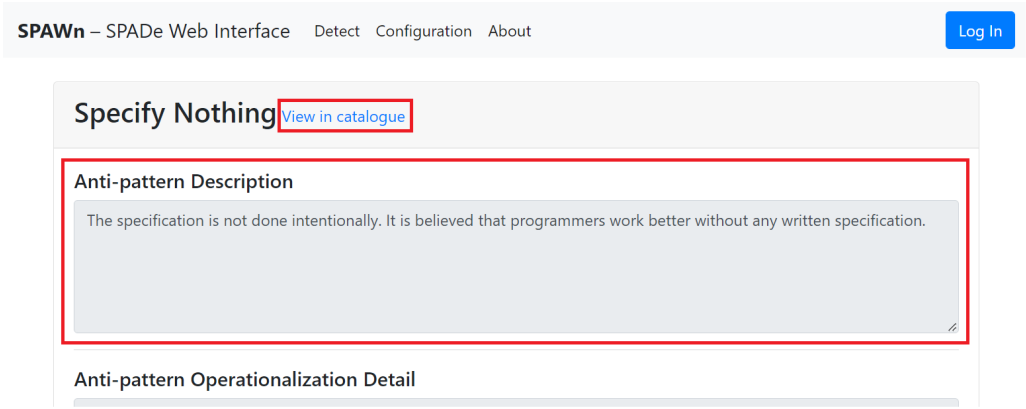
C.1 Zobrazení popisu anti-vzoru

Popis anti-vzoru je možné zobrazit na stránce jeho detailu. Stránka detailu se otevře z hlavní stránky klikem na název anti-vzoru ze seznamu všech dostupných anti-vzorů. Tabulka je zvýrazněna na obrázku C.1.



Obrázek C.1: Přechod na detail anti-vzoru

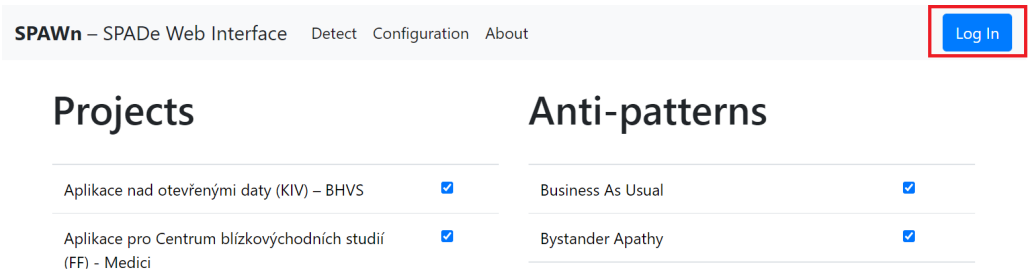
Popis se pak nachází v poli s názvem *Anti-pattern Description* (viz obrázek C.2). Některé anti-vzory mají svoji stránku v online katalogu, na kterou je možné přejít klikem na nápis *View in catalogue*, který je umístěn vedle názvu anti-vzoru (viz obrázek C.2). Pokud anti-vzor svoji stránku nemá, nápis bude přeškrtnutý.



Obrázek C.2: Popis anti-vzoru a odkaz do katalogu

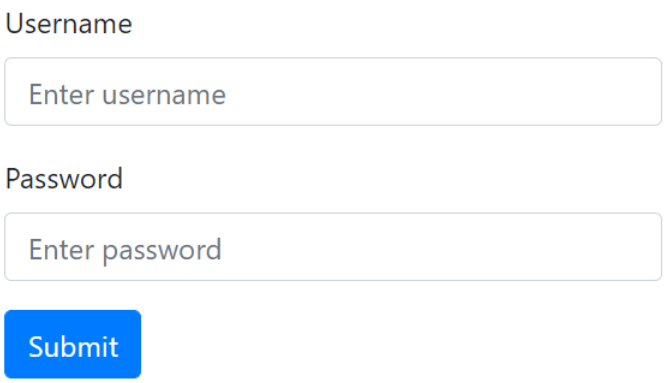
C.2 Přihlášení do aplikace

Přihlášení do aplikace se provede stiskem tlačítka *Log In*, které je umístěné v pravém horním rohu (viz obrázek C.3) v rámci horní lišty.



Obrázek C.3: Tlačítko pro přihlášení uživatele

Následně se zobrazí stránka s přihlašovacím formulářem (viz obrázek C.4), do kterého se vyplní jméno a heslo. Formulář se odešle stisknutím tlačíka *Submit*.

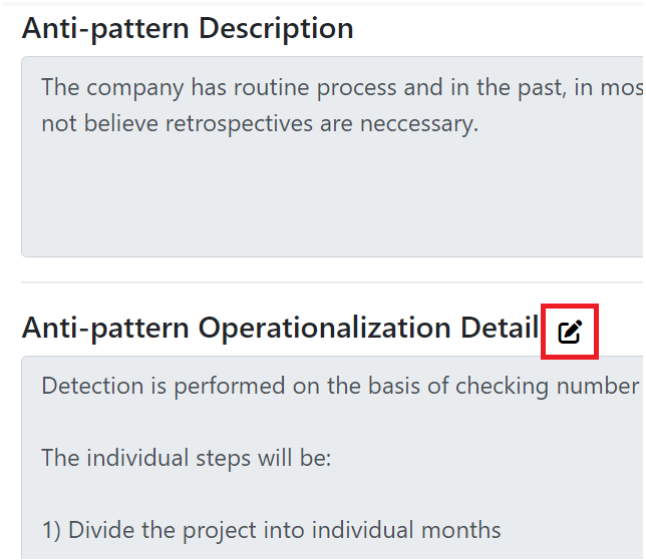


Obrázek C.4: Formulář pro zadání přihlašovacích údajů

Při zadání nesprávných údajů je uživatel upozorněn červeně podbarveným hlášením. V opačném případě je úspěšně přihlášen a přesměrován na hlavní stránku aplikace.

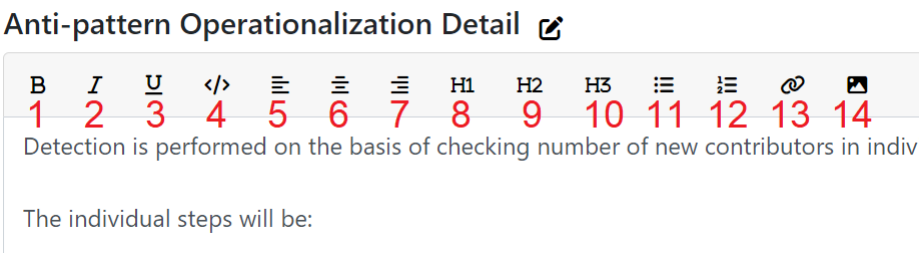
C.3 Editor operacionalizace

Popis operacionalizace anti-vzoru se nachází na stránce jeho detailu pod nadpisem *Anti-pattern Operationalization Detail*. Pro zahájení úpravy tohoto pole se stiskne tlačítko s motivem tužky (viz obrázek C.5). Editace je umožněna pouze přihlášeným uživatelům.



Obrázek C.5: Zapnutí režimu editace

Po stisknutí tlačítka editace se otevře editor (viz obrázek C.6).



Obrázek C.6: Prvky editoru operacionalizace

Tlačítka mají tyto funkce:

- **1** – tučné písmo,
- **2** – kurzíva,
- **3** – podtržení,
- **4** – formátování programového kódu,
- **5** – zarovnání textu doleva,
- **6** – zarovnání textu na střed,
- **7** – zarovnání textu doprava,
- **8** – nadpis stylu H1,
- **9** – nadpis stylu H2,
- **10** – nadpis stylu H3,
- **11** – vložení odrážkového seznamu,
- **12** – vložení číslovaného seznamu,
- **13** – vložení hypertextového odkazu,
- **14** – vložení obrázku.

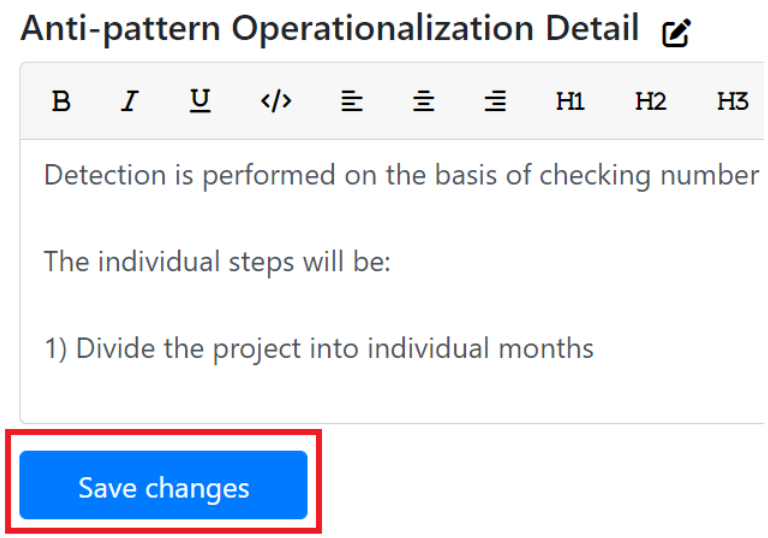
Při použití funkce formátování programového kódu není vybraná část textu formátovaná ihned. Nejprve dojde k obalení označeného textu tagem `<code>` a formátovaní je aplikováno až po uložení operacionalizace.

Vložení obrázku probíhá výběrem souboru s obrázkem ze zařízení uživatele uživatele. Maximální velikost obrázku je omezena na 1 MB a při jejím

překročení je uživatel upozorněn. Výška a šířka obrázku není v editoru upravitelná. Z toho důvodu musí být rozměry obrázku upraveny ještě před jeho vložením do pole popisu operacionalizace, popřípadě přímo v HTML souboru na serveru.

Úprava pole operacionalizace může probíhat dalšími dvěma způsby. Jedním z nich je formátování přímým použitím HTML tagů v poli. Dalším způsobem je přímá úprava HTML souborů s popisy operacionalizací na serveru s aplikací.

Uložení změn v poli operacionalizace se provede stisknutím tlačítka *Save changes* (viz obrázek C.7).

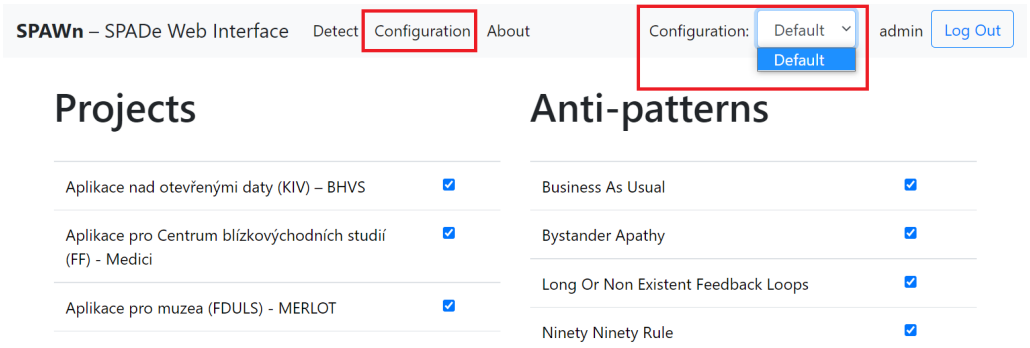


Obrázek C.7: Uložení popisu operacionaizace

C.4 Práce s konfiguracemi

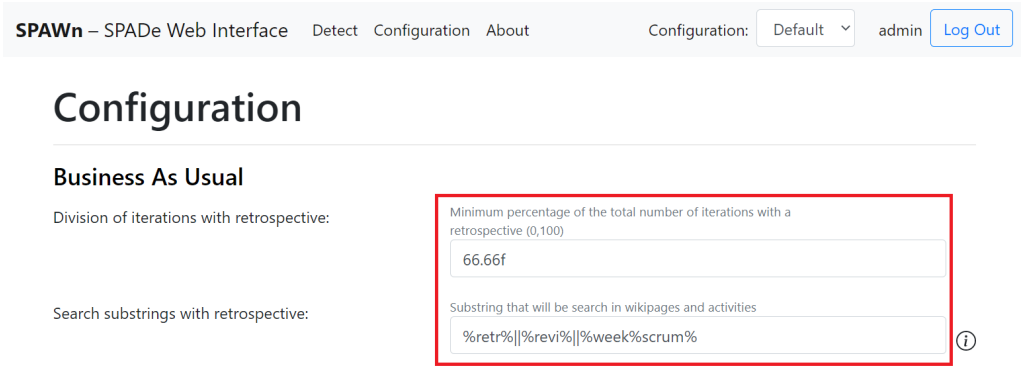
Aktuální konfigurace je zobrazena v horní liště aplikace u nápisu *Configuration*. Jinou konfiguraci je možné vybrat z rozevíracího seznamu, který obsahuje všechny uložené konfigurace (viz obrázek C.8). Tento seznam je dostupný pouze pro přihlášené uživatele. Nepřihlášení uživatelé mají k dispozici pouze výchozí konfiguraci.

Po vybrání konfigurace je uživatel přesměrován na stránku s nastavením prahových hodnot této konfigurace pro jednotlivé anti-vzory. Na tuto stránku je také možné se dostat stiskem tlačítka *Configuration* v hlavní navigaci.



Obrázek C.8: Výběr konfigurace

Na stránce konfigurace je možné libovolně měnit jednotlivé prahové hodnoty (viz obrázek C.9). Pro zobrazení prahových hodnot anti-vzoru je nutné rozkliknout pole s názvem anti-vzoru.

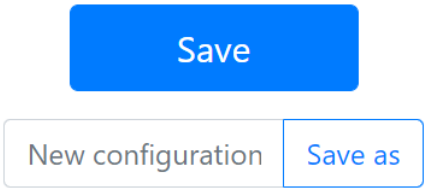


Obrázek C.9: Změna prahové hodnoty

Upravenou konfigurci je možné uložit, popřípadě uložit ji jako novou konfiguraci. K tomuto účelu se na konci stránky nacházejí dvě tlačítka (viz obrázek C.10). Tlačítko *Save* slouží k uložení změn do stávající konfigurace. Oproti tomu tlačítko *Save as* umožní uložení změn do nové konfigurace. Název nové konfigurace je uživatelem zadán do pole vedle tlačítka *Save as*. Nově uložená konfigurace se vloží do seznamu všech konfigurací a nastaví se jako právě vybraná.

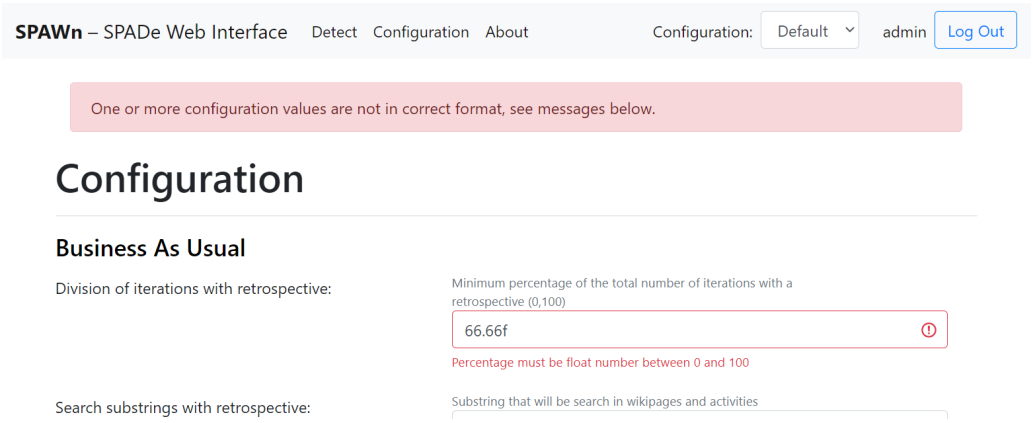
Možnosti úpravy prahových hodnot v konfiguracích jsou dostupné pouze pro přihlášené uživatele.

Ve výchozí konfiguraci s názvem *Default* není uživatelům umožněno prahové hodnoty měnit.



Obrázek C.10: Uložení konfigurace

V případě, že je některá z prahových hodnot zadána v nesprávném formátu, k uložení konfigurace nedojde a uživatel bude na chybu upozorněn (viz obrázek C.11).



Obrázek C.11: Upozornění na špatný formát prahové hodnoty

D Soubor pro popis anti-vzoru

V této příloze je uvedený příklad JSON souboru pro popis anti-vzoru. Z důvodu úspory místa jsou některé hodnoty zkrácené.

```
{
  "id": "9",
  "printName": "Bystander Apathy",
  "name": "BystanderApathy",
  "description": "The state in which individuals ...",
  "thresholds": [
    {
      "thresholdName": "searchSubstringsInvalid...",
      "thresholdType": "String",
      "thresholdPrintName": "Search substrings ...",
      "thresholdDescription": "Substring that ...",
      "thresholdErrorMess": "Maximum number of ..."
    },
    {
      "thresholdName": "maximumPercentageOfTasks...",
      "thresholdType": "Percentage",
      "thresholdPrintName": "Maximum percentage ...",
      "thresholdDescription": "Maximum ...",
      "thresholdErrorMess": "Percentage must ..."
    }
  ]
}
```