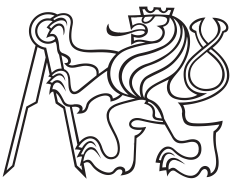
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Nástroj pro sledování změn uživatelského rozhraní webové aplikace

Pavla Kohákova

Květen 2016
Vedoucí práce: Ing. Miroslav Bureš, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Pavla Kováková**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Nástroj pro sledování a analýzu změn uživatelského rozhraní webové aplikace**

Pokyny pro vypracování:

Vytvořte nástroj, který bude procházet stránky představující uživatelské rozhraní webové aplikace a ukládat záznam stavu těchto stránek v konkrétním čase. K této části zadání můžete použít již existující modul. Uložené záznamy stránek bude pomocí nástroje možné navzájem srovnávat. K tomu bude nástroj nabízet sadu minimálně pěti srovnávacích technik založených na porovnávání zdrojových kódů stránek a jejich validity. Srovnání bude probíhat pro jednotlivé analyzované stránky i souhrnně pro celý záznam rozhraní webové aplikace. Nástroj bude výsledný report ukládat do databáze a zobrazovat. Implementovaný nástroj bude nezávislý na použitém prohlížeči webových stránek. Report bude zároveň možné exportovat do textového souboru. Implementovaný nástroj otestujte sadou vhodných funkcionálních testů.

Seznam odborné literatury:

Castro E.: HTML, XHTML a CSS. Computer Press, 2011
Bureš, M.: Metrics for Automated Testability of Web Applications. CompSysTech'15 - International Conference on Computer Systems and Technologies, ACM, 2015, 83-89

Vedoucí: Ing. Miroslav Bureš, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

prof. Ing. Filip Železný, Ph.D.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 6. 1. 2016

Poděkování / Prohlášení

Ráda bych poděkovala Ing. Miroslavu Burešovi, Ph.D. za cenné rady při vypracování bakalářské práce, za vstřícnost a čas, který mi věnoval při konzultacích této práce. Chci také poděkovat rodině za velkou podporu při studiu.

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

V Praze dne 20. května 2016

Abstrakt / Abstract

Cílem této bakalářské práce je navrhnout a implementovat nástroj, který bude schopen automaticky uložit zdrojový kód uživatelského rozhraní webové aplikace v několika časových verzích a tyto verze bude schopen porovnávat.

Nástroj tímto porovnáváním pomůže testerům webových aplikací analyzovat problémové úseky testované aplikace, jejíž strukturu nemusí předem znát. Nástroj analyzuje dynamické a problémové (např. nevalidní) části aplikace, které by jinak museli testeři zjišťovat manuálně, nebo kombinací více nástrojů.

Pro srovnávání verzí je v práci navrženo osm testovacích technik (testovací technika na počet všech elementů, na počet výskytů zadaného podřetězce, na počet výskytů zadaného elementu, na počet výskytů zadaného atributu, na stabilitu atributu, na počet validačních chyb a varování, na rozdílná a stejná místa v kódu, na výskyt vložených prvků).

Na základě analýzy požadavků byl nástroj naimplementován v jazyce Java. Nástroj analyzuje webové aplikace pomocí dvoufázového procesu - nejprve stáhne a uloží časové verze stránek a tyto poté analyzuje pomocí definovaných testovacích technik. Naimplementovaný nástroj byl otestován.

Klíčová slova: testování software; webová aplikace; dynamický web; automatizované testování.

The purpose of this thesis is to design and implement a tool for tracing changes in web application interface. The tool will be able to automatically download the source code of the application interface in different time intervals and then compare these versions.

The tool will help web application testers in analyzing the problem sections of tested application of which they do not need to know the structure in advance. The tool analyzes dynamic and problematic (e.g. invalid) parts of the application. Those parts would otherwise have to be examined manually by the testers or with a combination of multiple tools.

There are eight testing techniques proposed for version comparison in this thesis: a test technique for the total number of all elements, for the number of occurrences of a particular substring, for the number of occurrences of a particular element, for the number of occurrences of a particular attribute, for stability of a particular attribute, for the number of validation errors and warnings, for different and identical parts of the code, and for the detection of embedded objects.

Based on requirement analysis this tool was implemented in Java language. The tool analyzes the web applications in a two-step process. First it downloads and saves versions of the website in several time intervals. Then it analyzes them using the defined test techniques. The implemented tool has been tested.

Keywords: Software Testing, Web Applications, Dynamic Web, Automated Testing

Title translation: Tool for Tracking and Analysis of Changes in User Interface of Web Application

Obsah /

Kapitola 1
Úvod

Tvorba webových stránek a aplikací je dnes zcela běžnou součástí vytváření softwarových systémů. Kvalita vytvořené webové aplikace nebo jejích částí bývá testována z mnoha hledisek – automatickými či manuálními testy ověřují testeři, že aplikace funguje tak, jak bylo požadováno. Testeři ověřují validitu stránek, zobrazování v různých prohlížečích a na různých zařízeních (počítače, mobilní telefony, tablety a další), rychlost reakce aplikace na pokyny uživatele a další aspekty.

Kvalitu aplikace je možné testovat manuálně i automaticky. Automatické testy mají výhodu oproti manuálním testům tehdy, je-li aplikace příliš velká pro manuální testy, nebo pokud se testování často opakuje. Tvorba kvalitních automatických testů může být časově náročnější než jednorázové manuální testování, ale pokud je nutné aplikaci testovat častěji na stejné procesy, ušetří automatické testy firmám čas i peníze. Napsat dobrý test, který bude vždy znovupoužitelný, není však jednoduché.

Obtížnost vytváření automatického testu je závislá na struktuře webových stránek testované aplikace. Zdrojový kód stránek by správně měl být v souladu se standardy konsorcia W3C, v praxi se ovšem vyskytuje v kódech mnoho prohřešků proti pravidlům – některé elementy např. nemají požadované atributy, jiné nejsou uzavřené apod. Tyto chyby nemusí bránit ve správném zobrazování, protože prohlížeče umí obvykle zpracovat i nevalidní HTML kód, ale chyby ztěžují práci testerům.

Dalším aspektem, který znesnadňuje tvorbu automatických testů, je dynamika webové aplikace. Automatické testy se mnohdy opírají o statické prvky HTML kódu a aplikace, které generují obsah dynamicky (např. nabídka letů, internetové zpravodajství a mnoho dalších), nebo mění často hodnoty atributů (např. dynamicky generované stránky, jejichž elementům je přidělen vždy jiný identifikátor) jsou proto náročné pro psaní automatických testů.

Kvůli uvedeným komplikacím je nutné, aby testeři stránku analyzovali a určili všechny skutečnosti, které mohou ovlivnit jejich práci. Vývoj aplikace je totiž často od testování oddělen, a tak může být struktura aplikace testerům zcela neznámá a musejí rychle určit, jaké části aplikace jsou stabilní a jaké dynamické, a kde jsou v aplikaci problémová místa. Poté podle toho mohou zvolit nejvhodnější způsob, jakým danou část aplikace testovat.

Existují již nástroje, které jsou schopny analyzovat některé části kódu, validitu stránky apod., ale neporovnávají automaticky stránky v čase.

To se stalo motivací pro Nástroj pro sledování změn uživatelského rozhraní webové aplikace (dále jen Nástroj). Tento Nástroj by měl testerům usnadnit práci tím, že automaticky stáhne zdrojové kódy webové aplikace v několika různých časech a tyto stažené verze bude umět porovnávat a analyzovat několika technikami. Techniky budou zvoleny tak, aby jejich kombinací tester co nejrychleji určil stránky aplikace, které vyžadují zvýšenou pozornost. Techniky se zaměří na změny v těchto stránkách a na místa a předměty změn testera upozorní.

Tato implementační bakalářská práce je rozdělena do pěti kapitol: *Úvod*, *Analýza a návrh*, *Implementace*, *Testování* a *Závěr*.

V kapitole *Analýza a návrh* definuji do hloubky požadavky na funkce Nástroje. Definuji také testovací techniky, které budou v rámci Nástroje implementované. Tyto definice vychází především z odborných článků [1] a [2], které se problematikou metrik pro automatické testování zabývají.

V kapitole *Implementace* popisuji technické řešení Nástroje a věnuji se provedení a procesu implementace Nástroje.

V kapitole *Testování* se zabývám aktivitami, jimiž jsem ověřila správnost naimplementovaného Nástroje.

Tato práce se nezabývá detailně implementací modulu pro procházení stránek a jejich ukládání, protože se jedná o převzatý modul. Tvorba tohoto modulu byla součástí bakalářské práce *Nástroj pro hledání zadaných vzorů v uživatelském rozhraní webové aplikace*[3], ze které je modul převzatý.

Kapitola 2
Analýza a návrh

Před zahájením tvorby Nástroje bylo nutné nejprve detailně rozvrhnout, jaké funkce má Nástroj plnit a promyslet, jakým způsobem lze těchto funkcí dosáhnout. V této kapitole tedy definuji zásadní terminologii a specifikuji funkcionální a nefunkcionální požadavky. Připravím přehled funkcí a detailně rozeberu ty zásadní. Také zde definuji testovací techniky, pomocí kterých bude Nástroj schopen srovnávat stažené stránky. V rámci návrhu je také důležité stanovit vzhled a ovládání uživatelského rozhraní.

2.1 Terminologie

Nástroj – pojmem *Nástroj* je míněna vyvíjená aplikace *Nástroj pro sledování změn uživatelského rozhraní webové aplikace.*

Aplikace – pojmem *aplikace* je míněna jakákoli webová aplikace či doména, která bude zpracovávána pomocí Nástroje. V rámci Nástroje je tato aplikace identifikována jménem a uživatelským jménem. Jméno je název domény oproštěn o neunikátní řetězce jako např. protokol http – jméno je tedy např. seznam.cz, fel.cvut.cz aj. Uživatelské jméno identifikuje aplikaci proto, že je možné stahovat stránky aplikace i jako přihlášený uživatel, což značně ovlivňuje podobu zdrojového kódu (např. při přihlášeném procházení e-mailů). Pokud je uživatelské jméno *anonymous*, znamená to, že aplikace byla stažena bez jakéhokoli přihlašování.

Testování stránky, testovací technika – Jako *testovací technika* je označena analytická metoda, kterou jsou srovnávány zdrojové kódy stažené stránky ve dvou odlišných časových verzích. Proces srovnávání stránky testovacími technikami je označen jako *testování stránky.* Výrazy související s testováním jsou použity výhradně v souvislosti s těmito technikami, výjimkou je kapitola 4, kde se mluví o testování Nástroje. V kapitole Testování jsou však pojmy vázající se k testování Nástroje pozorně upřesněné tím, že mají konkretizující přídavné jméno (např. JUnit test), nebo jsou nahrazeny synonymy.

Konkrétní test – označuje testovací techniku s konkrétními parametry.

Testovací scénář, scénář – soubor několika konkrétních testů, které mají být provedeny v daném pořadí

Běh scénáře – provedené vykonání konkrétního testovacího scénáře pro konkrétní aplikaci a dvě konkrétní verze této aplikace.

Crawlování, crawler – Výraz crawling, počeštěný jako *crawlování*, je termín z oblasti informačních technologií označující automatizované stahování stránek z nějaké domény na základě zadání výchozího odkazu a následování odkazů na další stránky. *Crawler* je robot sloužící k tomuto stahování. Crawler přistupuje na stránky uživatelského rozhraní

webové aplikace, které vrací server, proto se nedostane na zdrojové soubory dynamického webu (například na PHP skripty).

2.2 Funkcionální požadavky

Nyní definuji všechny funkce a vlastnosti, které má výsledný Nástroj obsahovat. Funkcionální požadavky jsem rozdělila do tří skupin.

Ve skupině *Uložení stránek k analýze* definuji, jak má Nástroj umět nakládat se stahováním stránek a s crawlováním aplikací. Samotný crawler je převzatý, nicméně je nutné definovat možnosti jeho použití v Nástroji.

Ve skupině *Analýza uložených stránek pomocí definovaných testovacích technik* definuji procesy pro analyzování dvou verzí stránek nebo aplikace. Jednotlivé testovací techniky jsou rozvedeny v kapitole 2.6.

Ostatní funkcionální požadavky jsou uvedeny ve skupině *Další požadavky*.

2.2.1 Uložení stránek k analýze

1. **Vytvoření konfiguračního souboru pro crawlování.** Uživatel bude moci pomocí grafického rozhraní vytvořit konfigurační soubor, ve kterém bude uloženo nastavení pro crawlování. Budou existovat tři možnosti konfiguračního souboru – pro anonymní crawlování aplikace, pro přihlášené crawlování aplikace a pro stažení seznamu stránek.
2. **Smazání konfiguračního souboru.** Uživatel bude moci smazat jakýkoli konfigurační soubor pro crawlování.
3. **Crawlování webové aplikace na základě konfiguračního souboru.** Uživatel bude moci spustit proces crawlování na základě výběru konfiguračního souboru pro crawlování aplikace. Uživatel může proces crawlování pozastavit a znovu spustit, případně úplně zrušit.
4. **Stažení sady stránek na základě konfiguračního souboru.** Uživatel bude moci spustit stahování stránek na základě výběru konfiguračního souboru se seznamem stránek.
5. **Přehledné zobrazení stažených aplikací včetně stromu stránek a zobrazení verzí, v nichž byly aplikace staženy.** Nástroj bude zobrazovat seznam stažených aplikací a seznam verzí, v nichž byly aplikace staženy. Nástroj zobrazí seznam stránek aplikace ve zvolené verzi.
6. **Smazání stažené aplikace.** Uživatel bude moci smazat celou staženou aplikaci.
7. **Smazání starých verzí aplikace.** Uživatel bude moci snadno odstranit staré verze aplikace, s nimiž již nechce pracovat.
8. **Zobrazení zdrojového kódu stažené stránky.** Uživatel bude moci zobrazit zdrojový kód stránky, zvolí-li verzi stránky.

2.2.2 Analýza uložených stránek pomocí definovaných testovacích technik

9. **Vytvoření testovacího scénáře.** Uživatel bude moci vytvořit scénář, který bude obsahovat posloupnost uživatelem definovaných konkrétních testů.
10. **Zobrazit testovací scénáře.** Nástroj bude zobrazovat vytvořené testovací scénáře a jejich strukturu.
11. **Úprava testovacích scénářů.** Uživatel bude moci scénáře přejmenovat, označit je jako neaktivní či aktivní, aby se ne/zobrazovaly v nabídce scénářů. Uživatel bude moci měnit pořadí konkrétních testů ve scénáři. Uživatel bude moci přidávat či mazat konkrétní testy.

12. **Smazání testovacího scénáře.** Uživatel bude moci odstranit scénář, který již nechce používat.
13. **Testování dvou odlišných verzí stránky testovacím scénářem.** Uživatel bude moci srovnat dvě verze libovolné stránky libovolným testovacím scénářem.
14. **Testování všech stránek v aplikaci testovacím scénářem a vytvoření souhrnné zprávy o testování celé aplikace**. Uživatel bude moci analyzovat dvě verze libovolné aplikace libovolným testovacím scénářem. Nástroj zobrazí uživateli souhrnnou zprávu o testování.
15. **Zobrazení zpráv z již proběhlých testování.** Uživatel bude moci zobrazit výsledky dříve proběhlých testování pro libovolnou aplikaci či stránku.
16. **Uložení výsledné zprávy o testování do textového souboru.** Uživatel bude moci uložit zobrazenou zprávu do textového souboru do libovolné složky.

2.2.3 Další požadavky

17. **Kontrola nechtěného zavření okna, v němž probíhá práce.** Nástroj bude upozorňovat uživatele, pokud bude uživatel chtít zavřít okno, ve kterém právě probíhá některý z důležitých procesů Nástroje.
18. **Volba českého či anglického jazyka.** Nástroj bude lokalizován do českého a anglického jazyka s možností rozšířit Nástroj později o další verze.

2.3 Nefunkcionální požadavky

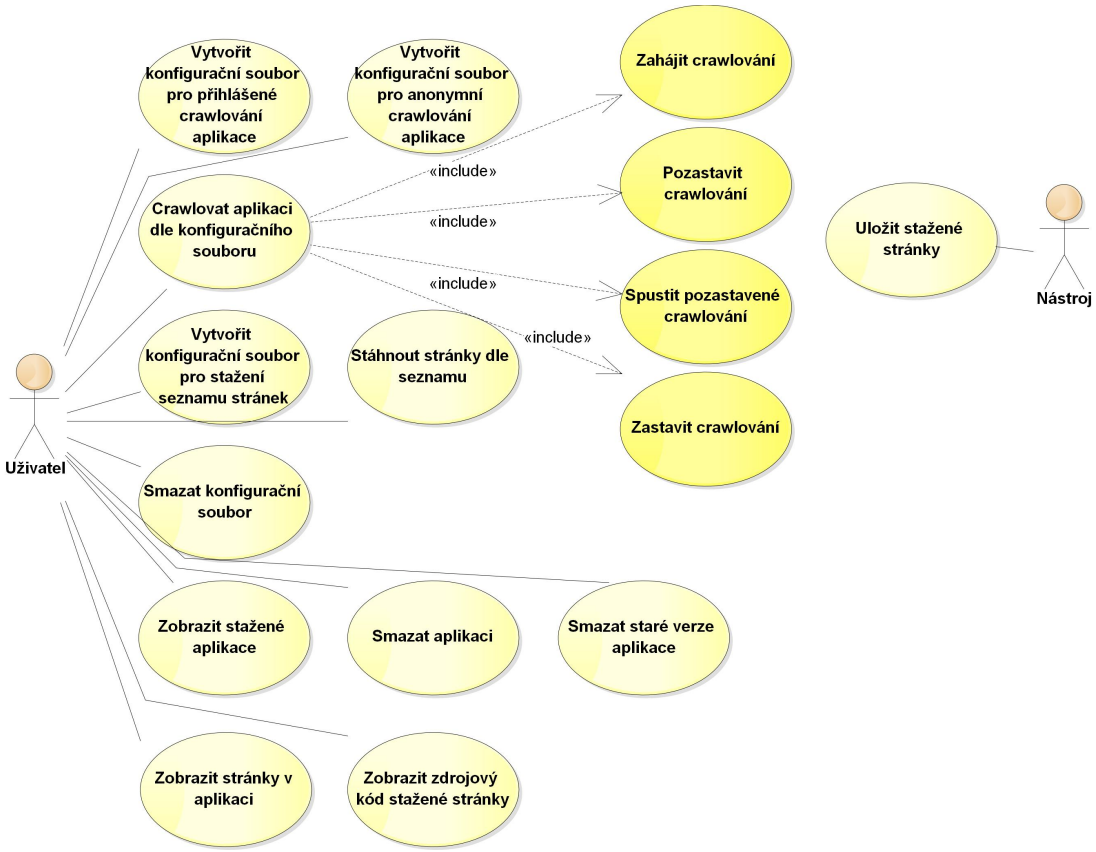
Zde bych ráda uvedla požadavky, které se netýkají funkcionalit Nástroje, ale spíše technického provedení. Mezi nefunkcionální požadavky se počítají např. nároky na architekturu aplikace, na výkon, na bezpečnost, design, licenci, výrobní cenu apod. Pro Nástroj jsem definovala následující nefunkcionální požadavky:

1. **Desktopová aplikace.** Nástroj bude implementovaný jako desktopová aplikace, vše bude spustitelné na jednom počítači.
2. **Implementace v jazyce Java.** Nástroj bude kompletně naimplementován v programovacím jazyce Java. Nástroj může využívat cizí knihovny, které jsou k volnému použití.
3. **Přenositelnost.** Nástroj bude nezávislý na operačním systému.
4. **Rozšiřitelnost kódu.** Kód bude možné rozšířit o další funkce a bude možné upravovat funkce stávající.
5. **Dokumentace.** Kód bude obsahovat dokumentaci v anglickém jazyce, aby se mohli pokročilí uživatelé snadněji zorientovat a do kódu zasahovat.
6. **Možnost přidat další testovací techniky.** Nástroj bude možné obohatit o další testovací techniky pro srovnávání stránek.
7. **Možnost uživatelské konfigurace.** Uživatel bude moci sám nakonfigurovat, v jakém jazyce má být Nástroj spuštěn a bude moci zvolit výchozí úložiště. Uživatel také může sám konfigurovat parametry testovacích technik.
8. **Open source.** Zdrojové kódy Nástroje a dokumenty o Nástroji budou dostupné pro volné užití. Nástroj bude možné libovolně upravovat.

2.4 Případy užití

Z přehledu funkcionálních požadavků lze odvodit přehled případů užití. Případy užití zachycují akce, které konají různí aktéři při používání Nástroje. Každý případ užití se vztahuje alespoň k jednomu funkcionálnímu požadavku. **Za názvem uvádím v závorkách identifikátor příslušných funkcionálních požadavků,** které jsou implementované daným případem užití. Rozeberu případy užití rozdělené do dvou skupin tak, aby odpovídaly funkcionálním požadavkům pro crawlování a testování.

2.4.1 Uložení stránek k analýze

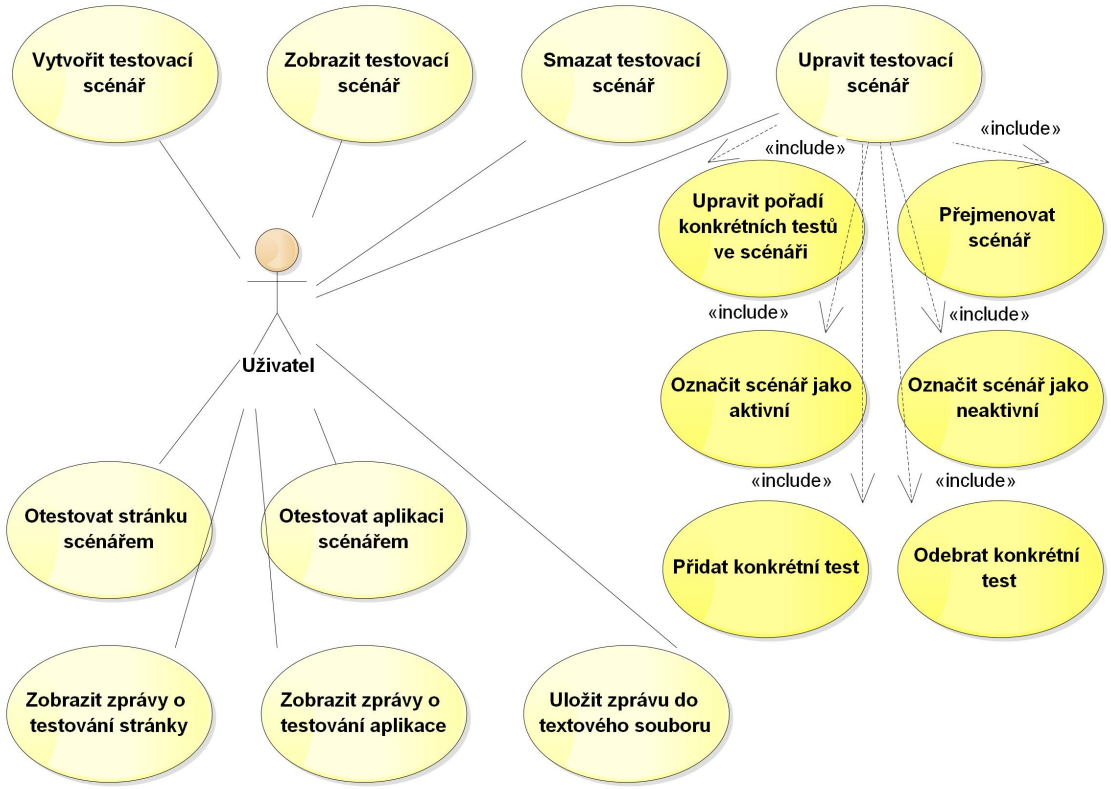


Obrázek 2.1. Diagram případů užití pro crawlování

- **Vytvořit konfigurační soubor pro anonymní crawlování aplikace** (1) – Uživatel vytvoří konfigurační soubor, v němž bude odkaz na výchozí stránku aplikace, která má být celá stažena, a to anonymně, bez přihlašování.
- **Vytvořit konfigurační soubor pro přihlášené crawlování aplikace** (1) - Uživatel vytvoří konfigurační soubor, v němž bude odkaz na výchozí stránku aplikace a údaje pro přihlášení uživatele, jako např. uživatelské jméno, heslo a vstupy pro tyto údaje na výchozí stránce.
- **Vytvořit konfigurační soubor pro stažení seznamu stránek** (1) - Uživatel vytvoří konfigurační soubor, který bude obsahovat sadu stránek, které se mají jednotlivě stáhnout. V souboru bude seznam adres stránek, které se budou stahovat bez přihlášení a bude stažena jen stránka ze zadané adresy, nikoli celá aplikace přístupná z adresy.

- **Smazat konfigurační soubor** (2) – Uživatel může smazat konfigurační soubor. Nezáleží, zda vybere soubor pro anonymní crawlování aplikace, soubor pro přihlášené crawlování aplikace, nebo soubor pro stažení seznamu stránek.
- **Crawlovat aplikaci dle konfiguračního souboru** (3) - Uživateli bude stačit, zvolí-li konfigurační soubor, který obsahuje výchozí URL pro stažení aplikace a případně přihlašovací údaje, a bude moci proces crawlování spustit. Tento proces může uživatel pozastavit a znovu spustit, případně úplně zrušit. Uživatel bude průběžně informován pomocí uživatelského rozhraní o průběhu a výsledku crawlování. Uživatel bude moci spustit více procesů crawlování najednou, s jedinou limitací, že nebude možné spustit crawlování stejné aplikace dvakrát ve stejné minutě. Toto omezení je kvůli přehlednosti aplikace, bude však možné upravit kód snadno tak, aby pracoval s verzemi aplikace nejen s přesností na minuty, ale s přesností na sekundy.
- **Zahájit crawlování** (3) – Uživatel zahájí proces crawlování dle zvoleného konfiguračního souboru.
- **Pozastavit crawlování** (3) – Uživatel pozastaví spuštěné crawlování. Běh crawlování může být obnoven.
- **Spustit pozastavené crawlování** (3) – Uživatel opět spustí crawlování, které bylo pozastavené.
- **Zastavit crawlování** (3) – Uživatel zastaví crawlování. Zastavený běh crawlování již nebude možné znovu spustit.
- **Stáhnout stránky dle seznamu** (4) – Uživatel zvolí konfigurační soubor se sadou stránek, a Nástroj stáhne postupně všechny jednotlivé stránky ze sady. Bude-li stránka stažena víckrát v jedné minutě, bude uložena pouze aktuálnější verze. Kód bude možné snadno upravit pro stažení verze s přesností na vteřiny, nejen minuty. Tato funkce umožní testerům soustředit se na stránky, o kterých ví, že jsou problematické, aniž by museli stahovat znovu všechny stránky aplikace.
- **Uložit stažené stránky** (3, 4) – Nástroj uloží do datového úložiště stránky, které crawler stáhl.
- **Zobrazit stažené aplikace** (5) - Nástroj bude zobrazovat seznam stažených aplikací (jméno aplikace a uživatelské jméno, pod kterým bylo provedeno přihlášení). Ke každé aplikaci bude uveden seznam verzí, kdy byla aplikace crawlována. Zvolí-li uživatel verzi, zobrazí se mu strom stránek, které byly v této verzi staženy.
- **Zobrazit stránky v aplikaci** (5) – Uživatel zvolí aplikaci a její verzi. Nástroj zobrazí strom stránek, které byly v této verzi staženy.
- **Smazat aplikaci** (6) - Uživatel bude moci smazat celou staženou aplikaci – včetně zdrojových kódů všech verzí a výsledků testů, které pro tuto aplikaci proběhly.
- **Smazat staré verze aplikace** (7) - Uživatel bude moci odstranit staré verze aplikace, které již nepotřebuje porovnávat. Uživatel vybere datum verze, kterou chce zachovat a všechny starší verze budou odstraněny, včetně zdrojových kódů a výsledků testů, které proběhly pro tyto verze.
- **Zobrazit zdrojový kód stažené stránky** (8) - Zvolí-li uživatel nějakou verzi a stránku v této verzi, zobrazí se mu zdrojový kód stránky. Uživatel bude upozorněn, pokud zvolená stránka není stránkou, ale pouze složkou, tedy rozcestníkem k dalším stránkám stažené aplikace.

2.4.2 Analýza uložených stránek pomocí definovaných testovacích technik



Obrázek 2.2. Diagram případů užití pro testování

- **Vytvořit testovací scénář** (9) - Uživatel bude moci vytvořit scénář, který bude obsahovat posloupnost konkrétních testů. Konkrétní testy vytvoří uživatel tím, že bude při tvorbě scénáře vybírat z předem definovaných technik a bude pouze doplňovat konkrétní hodnoty parametrů. Každý konkrétní test bude ve scénáři uložen pouze jednou.
- **Zobrazit testovací scénář** (10) – Uživatel bude moci zobrazit testovací scénář a konkrétní testy, které scénář obsahuje.
- **Upravit testovací scénář** (11) – Uživatel bude moci upravovat vytvořené scénáře.
- **Upravit pořadí konkrétních testů ve scénáři** (11) - Uživatel bude moci upravovat pořadí, ve kterém se budou při spuštění scénáře konkrétní testy vykonávat.
- **Přejmenovat scénář** (11) – Uživatel může scénář přejmenovat na jakékoli jméno, které se v nabídce scénářů dosud nevyskytuje.
- **Označit scénář jako neaktivní** (11) - Scénář bude možné označit jako neaktivní. Poté se již nebude dále zobrazovat v nabídce scénářů, které mohou být spuštěny, ale výsledky tohoto scénáře zůstanou zachovány.
- **Označit scénář jako aktivní** (11) - Scénář, který je označen jako neaktivní, bude možné vždy označit znovu jako aktivní. Poté bude scénář zobrazen v nabídce scénářů ke spuštění.
- **Přidat konkrétní test** (11) - Uživatel bude moci přidat do scénáře konkrétní test, který se ve scénáři ještě nevyskytuje. Běhy scénářů, které již proběhly, nebudou doplněny o výsledky tohoto konkrétního testu, konkrétní test však bude použit při dalších testováních, nebo při opětovném provedení stejného běhu.

- **Odebrat konkrétní test** (11) - Uživatel bude moci odebrat ze scénáře libovolný konkrétní test, kromě posledního zbývajícího testu ve scénáři. Pokud se konkrétní test vztahoval jen ke scénáři, ze kterého je právě odebrán, bude z Nástroje zcela smazán včetně výsledků testování.
- **Smazat testovací scénář** (12) - Pokud se uživatel rozhodne, že už nechce vytvořený scénář nikdy používat a chce smazat i výsledky spojené s testovacím scénářem, může scénář odstranit. Nástroj smaže scénář, konkrétní testy, které se vztahovaly pouze k tomuto scénáři a výsledky těchto testů pro všechny aplikace.
- **Otestovat stránku scénářem** (13) - Uživatel může zvolit libovolnou stránku v libovolné aplikaci a analyzovat rozdíly mezi dvěma verzemi této stránky libovolným testovacím scénářem. Výsledky analýzy budou zobrazeny v uživatelském rozhraní a uloženy pro další použití.
- **Otestovat aplikaci scénářem** (14) - Uživatel může vybrat aplikaci a analyzovat dvě její verze libovolným testovacím scénářem. Pro každý konkrétní test ze scénáře proběhne analýza všech jednotlivých stránek a na závěr bude pro každý konkrétní test ze scénáře zobrazena zpráva o stavu celé aplikace. V uživatelském rozhraní budou zobrazeny nejprve výsledky pro scénář pro celou aplikaci a poté výsledky pro jednotlivé stránky.
- **Zobrazit zprávy o testování aplikace** (15) - Nástroj bude pro každou aplikaci zaznamenávat běhy scénářů. Zvolí-li uživatel nějaký běh, zobrazí se souhrnné výsledky pro aplikaci pro tento scénář a testované verze.
- **Zobrazit zprávy o testování stránky** (15) – Uživatel může zvolit libovolnou stránku ze seznamu stránek pro konkrétní běh aplikace. Nástroj zobrazí výsledky všech konkrétních testů, kterými byla tato stránka analyzována.
- **Uložit zprávu do textového souboru** (16) – Uživatel bude moci uložit jakoukoli zprávu z hlavního okna do textového souboru.

2.5 Základní procesy v Nástroji

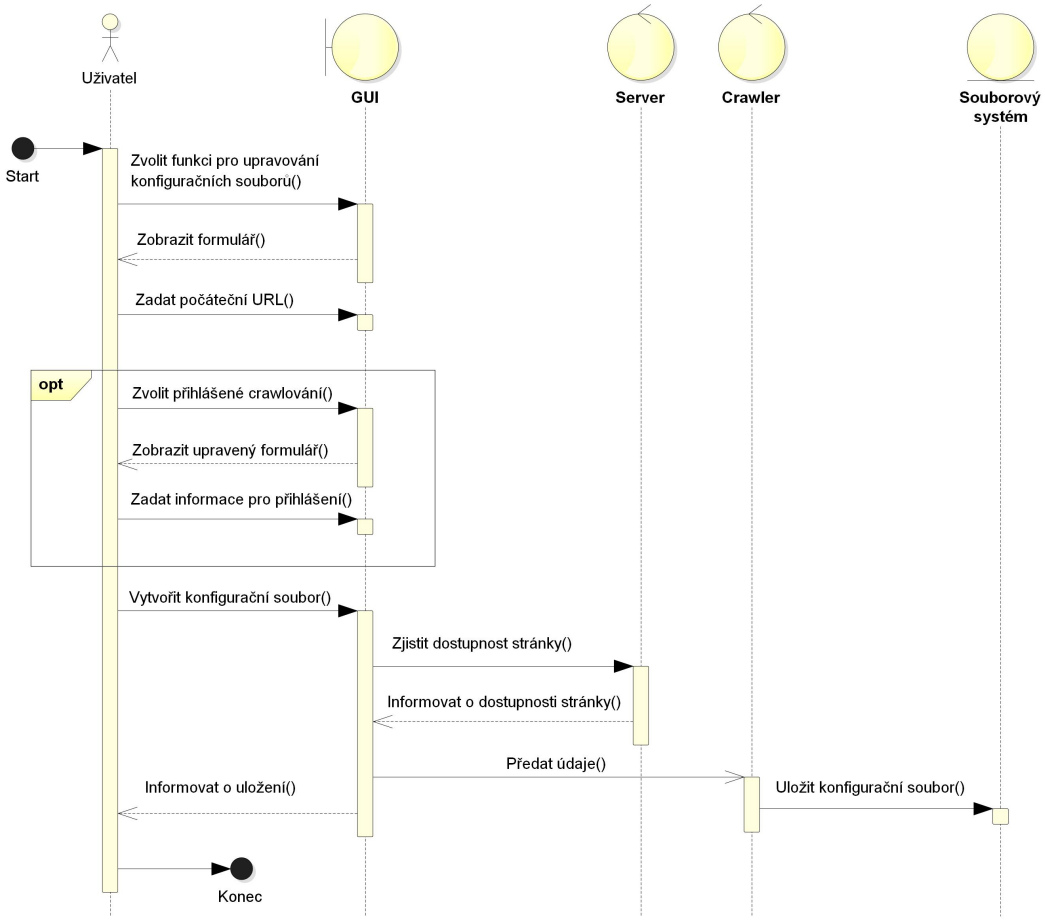
Nástroj bude mít mnoho funkcí, které jsem popsala výše. Pořadí, v jakém bude uživatel funkce využívat, je různé. Nyní popíši základní procesy, jakými uživatel používá Nástroj. Tyto procesy popíši formou scénáře, sekvenčního diagramu, či diagramu aktivit, dle charakteru procesu.

2.5.1 Vytvoření konfiguračního souboru pro crawlování aplikace

Diagram 2.3 zachycuje proces tvorby nového konfiguračního souboru pro crawlování celé aplikace na základě výchozího URL. Uživatelské rozhraní nabídne uživateli vstupy pro výchozí URL a v případě přihlášeného crawlování pro další informace, jako např. uživatelské jméno, heslo, XPathový přístup k přihlašovacímu formuláři a odkaz na odhlášení, který crawler nebude navštěvovat.

Poté, co uživatel zažádá o vytvoření konfiguračního souboru, zkontroluje uživatelské rozhraní, jestli nejsou prázdné vstupy, a také zkontroluje dostupnost počátečního URL, protože jde o nejdůležitější část konfiguračního souboru, bez které by nešlo aplikaci stáhnout. Správnost uživatelského jména a hesla musí hlídat sám uživatel.

Pokud jsou všechny vstupy vytvořené správně, předá je uživatelské rozhraní komponentě pro crawlování, která vytvoří nový konfigurační soubor a uloží jej do datového úložiště. Uživatel je o vytvoření informován.

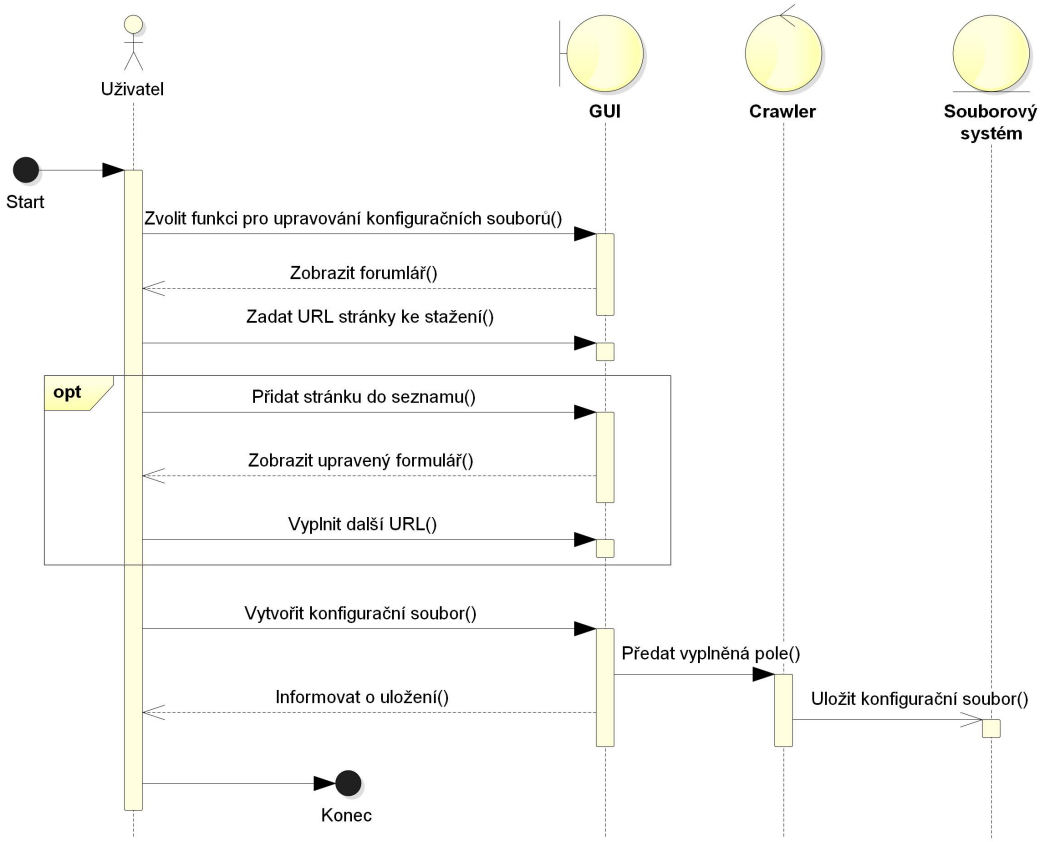


Obrázek 2.3. Sekvenční diagram vytváření konfiguračního souboru pro crawlování aplikace

2.5.2 Vytvoření konfiguračního souboru pro stažení sady stránek

Diagram 2.4 zachycuje proces tvorby nového konfiguračního souboru pro stahování sady stránek na základě seznamu URL. V uživatelském rozhraní se zobrazí formulář s polem pro zadání URL, uživatel může přidat libovolný počet stránek. Když uživatel požádá o vytvoření konfiguračního souboru, zkontroluje rozhraní, zda bylo vyplněné alespoň jedno URL a poté předá obsah vyplněných políček formuláře komponentě pro crawlování, která vytvoří konfigurační soubor a uloží jej do datového úložiště.

V tomto případě nebude kontrolována validita URL, protože URL zde nemá tak zásadní roli jako při tvorbě konfiguračního souboru pro crawlování aplikace. Uživatel bude na chybné URL upozorněn při procesu stahování sady stránek.



Obrázek 2.4. Sekvenční diagram vytvoření konfiguračního souboru na stažení sady stránek

2.5.3 Crawlování aplikace

Následující scénář zachycuje průběh crawlování aplikace. Jedná se o tzv. šťastný scénář, kdy vše probíhá, jak má. Tento scénář tedy nezachycuje případy výjimek, kdy v průběhu crawlování dojde k nějakým chybám (např. kdyby uživatelovo zařízení nemělo přístup k internetu, kdyby crawler nemohl načíst stránku, kdyby se nezdařilo přihlašování apod.). V případě chyby o ní bude uživatel upozorněn prostřednictvím uživatelského rozhraní.

1. Uživatel zvolí funkci pro crawlování.
2. Nástroj zobrazí nabídku konfiguračních souborů.
3. Uživatel zvolí z nabídky konfigurační soubor pro aplikaci, již chce stáhnout. Pokud konfigurační soubor ještě neexistuje, uživatel ho vytvoří. Poté uživatel spustí crawlování aplikace.
4. Nástroj načte a zpracuje konfigurační soubor. Nástroj vytvoří robota pro crawlování dle konfiguračního souboru a robota spustí. Postupně informuje uživatele o stažených stránkách a ukládá jejich zdrojové kódy. Nástroj poté informuje uživatele o skončení procesu crawlování.

2.5.4 Stažení sady stránek

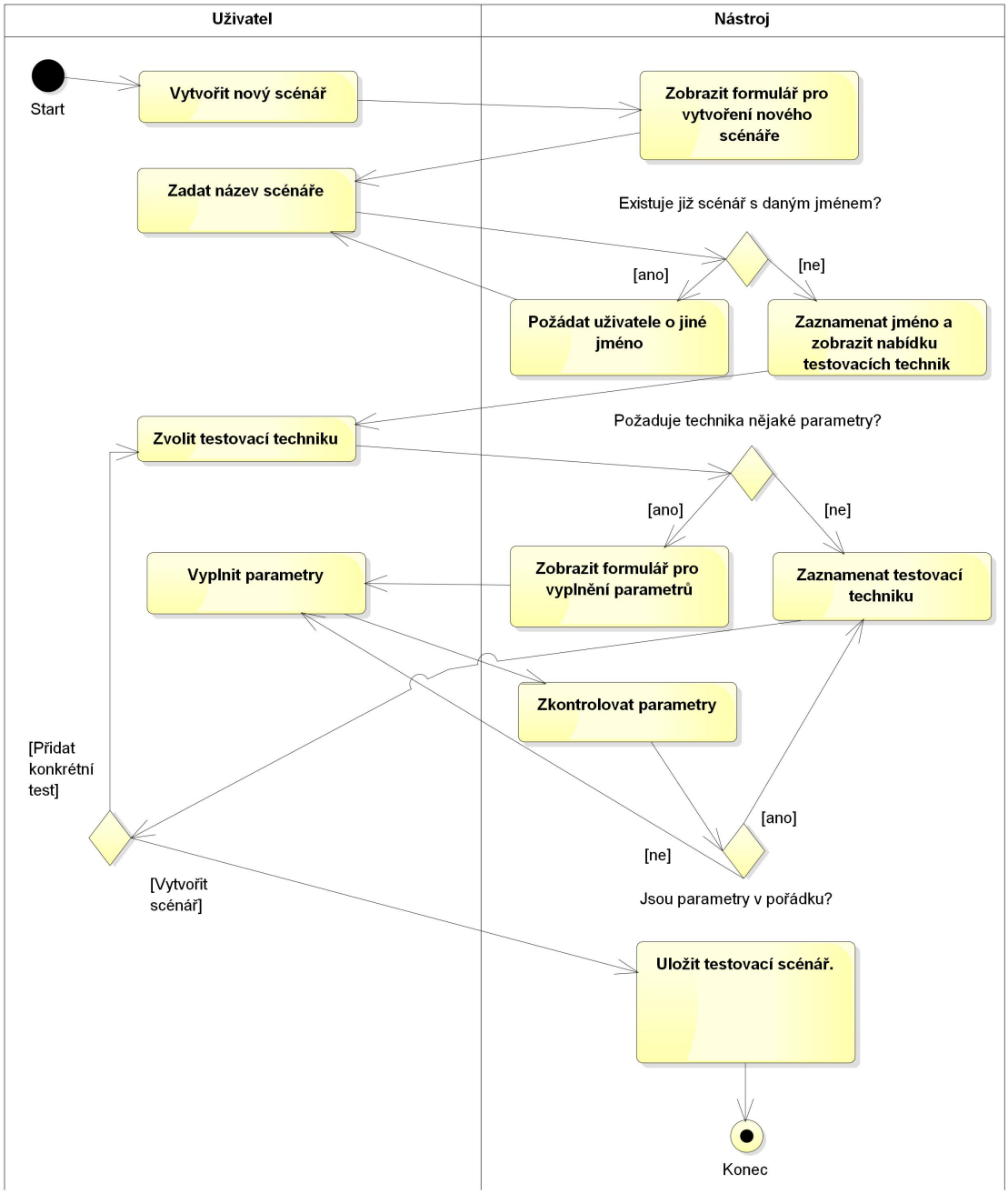
Následující scénář zachycuje průběh stahování sady stránek:

1. Uživatel zvolí funkci pro crawlování.
2. Nástroj zobrazí nabídku konfiguračních souborů.
3. Uživatel zvolí z nabídky konfigurační soubor, který obsahuje sadu stránek, kterou chce stáhnout. Pokud konfigurační soubor ještě neexistuje, uživatel ho vytvoří. Poté uživatel spustí stahování.
4. Nástroj načte konfigurační soubor. Postupně čte odkazy na jednotlivé stránky, které stahuje a ukládá jejich zdrojový kód. Nástroj informuje uživatele o úspěšně i neúspěšně stažených stránkách. Nakonec informuje uživatele o skončení procesu stahování sady stránek.

2.5.5 Vytvoření testovacího scénáře

Diagram 2.5 zachycuje proces vytváření testovacího scénáře. Uživatel může do scénáře přidat libovolný počet konkrétních testů, které budou uloženy v pořadí, ve kterém je uživatel přidával. Toto pořadí bude možné později měnit. Při vytváření scénáře odstraní Nástroj duplicitní konkrétní testy.

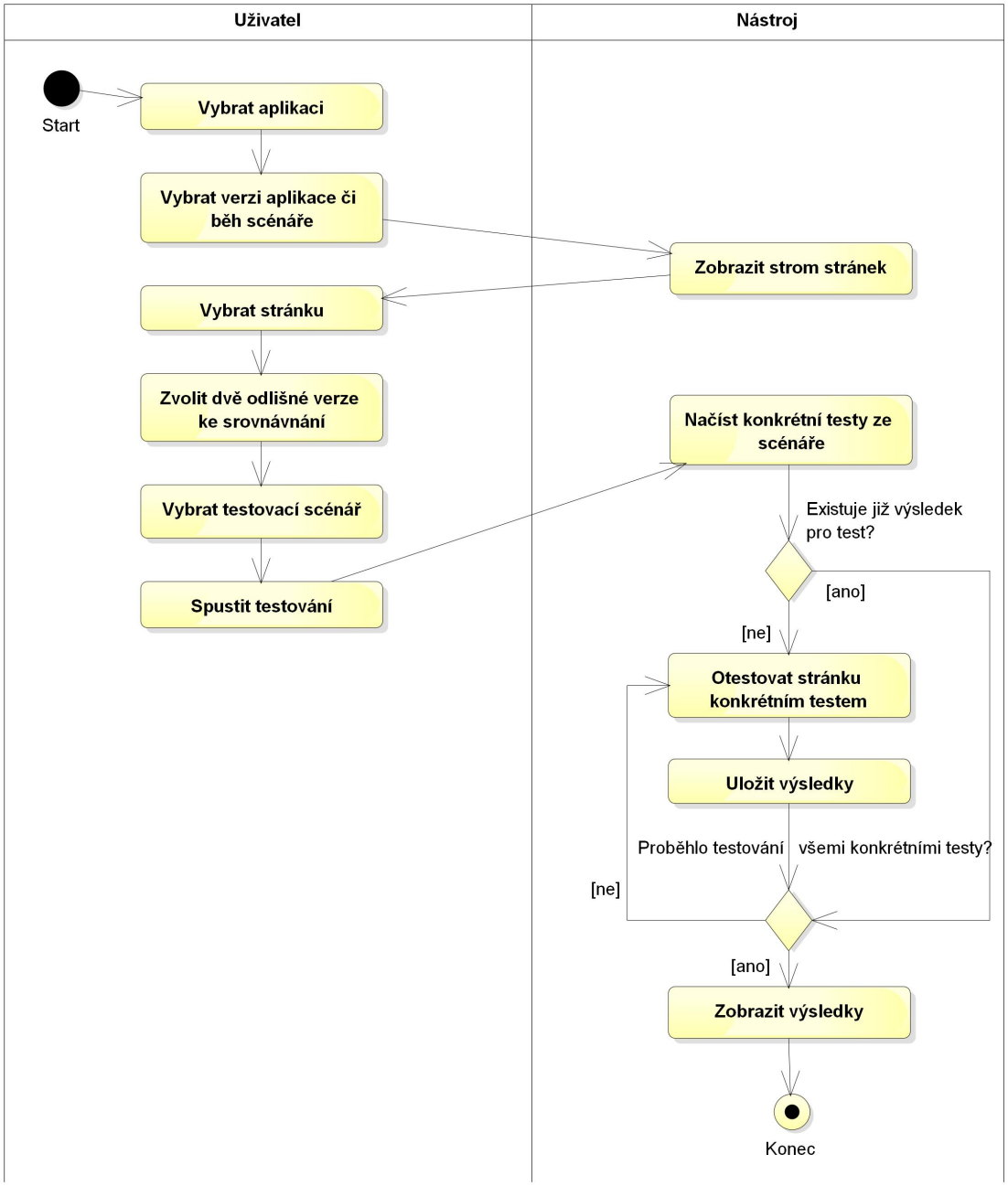
Poté, co uživatel zvolí testovací techniku, kterou chce přidat, zobrazí Nástroj formulář pro vyplnění parametrů této techniky a po odeslání zkontroluje Nástroj správnost vyplněných parametrů (např. zda uživatel nenechal pole prázdné, zda vyplnil číslici, když to technika vyžaduje apod.). O špatném vyplnění informuje Nástroj uživatele pomocí uživatelského rozhraní.



Obrázek 2.5. Diagram aktivit pro vytvoření testovacího scénáře

2.5.6 Testování stránky

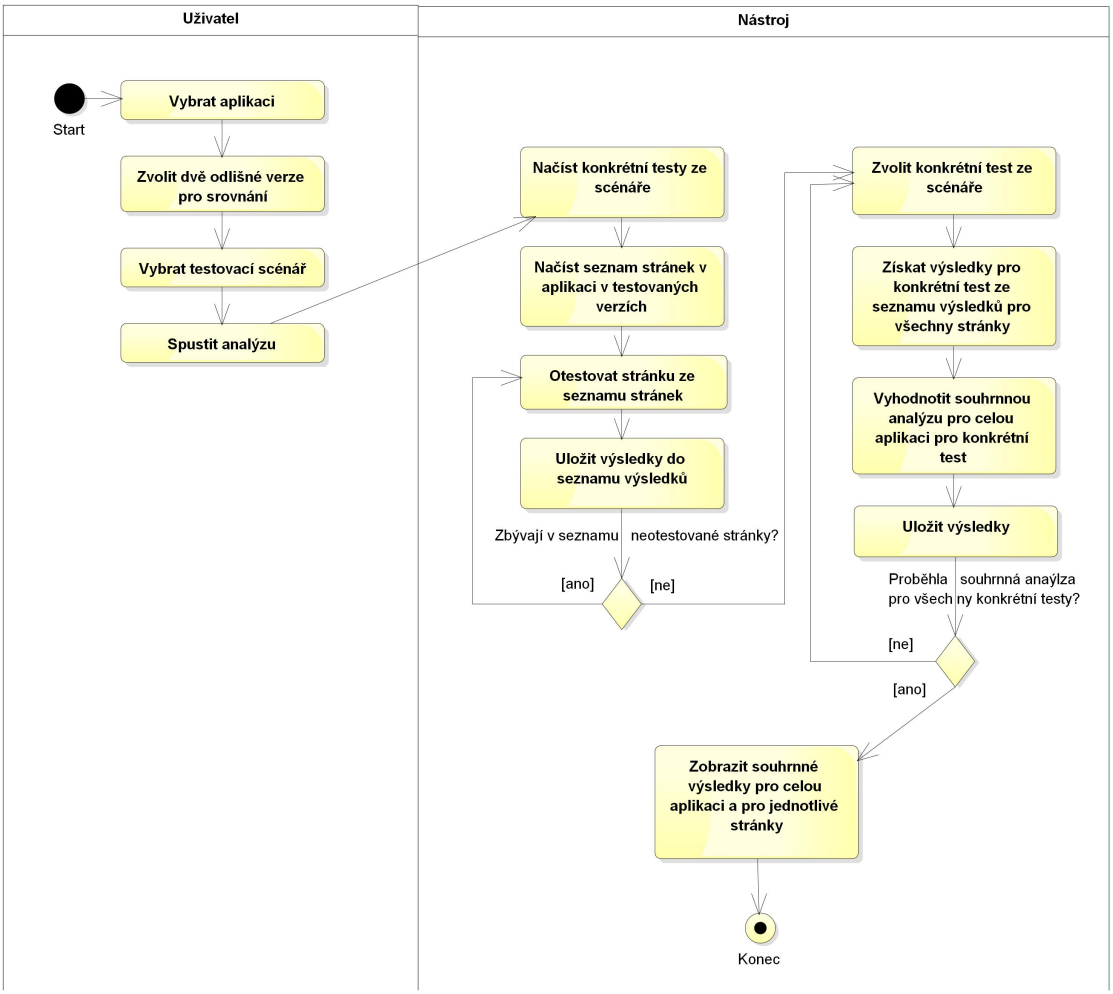
Diagram 2.6 zachycuje proces testování stránky. Uživatel zvolí stránku, dvě verze k porovnání a scénář. Nástroj se postará o otestování. Nástroj otestuje stránku postupně na všechny konkrétní testy ze scénáře. Souhrnné výsledky poté Nástroj zobrazí v hlavním okně uživatelského rozraní. Pokud byla stránka někdy dříve testována některým konkrétním testem ze scénáře, načte Nástroj výsledek tohoto testování z datového úložiště a urychlí tak proces testování.



Obrázek 2.6. Diagram aktivit pro testování stránky

2.5.7 Testování aplikace

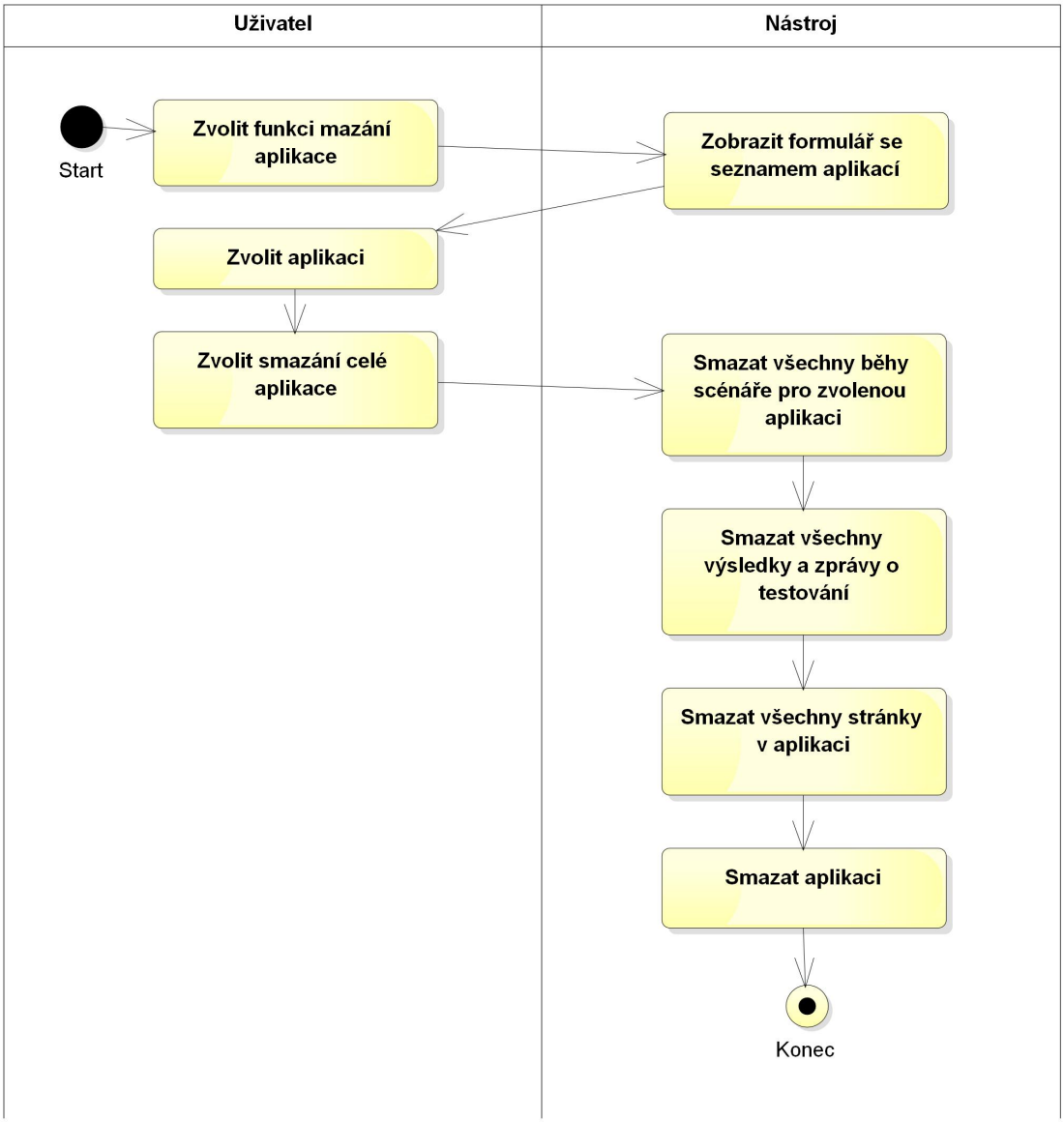
Diagram 2.7 zachycuje proces testování celé aplikace. Uživatel zvolí aplikaci, dvě verze k porovnání a scénář. Nástroj poté otestuje každou stránku konkrétními testy ze scénáře. Proces testování stránky je popsán v sekci 2.5.6. Výsledky testování stránek si Nástroj ukládá do seznamu. Poté, co jsou otestovány všechny stránky v aplikaci, zpracuje Nástroj výsledky pro jednotlivé stránky a provede pro každý konkrétní test souhrnnou analýzu pro celou aplikaci. Souhrnný výsledek testování zobrazí Nástroj v hlavním okně uživatelského rozhraní – nejprve zobrazí výsledky pro všechny konkrétní testy ze scénáře pro celou aplikaci, poté zobrazí výsledky pro každou stránku z aplikace.



Obrázek 2.7. Diagram aktivit pro testování aplikace

2.5.8 Smazání aplikace

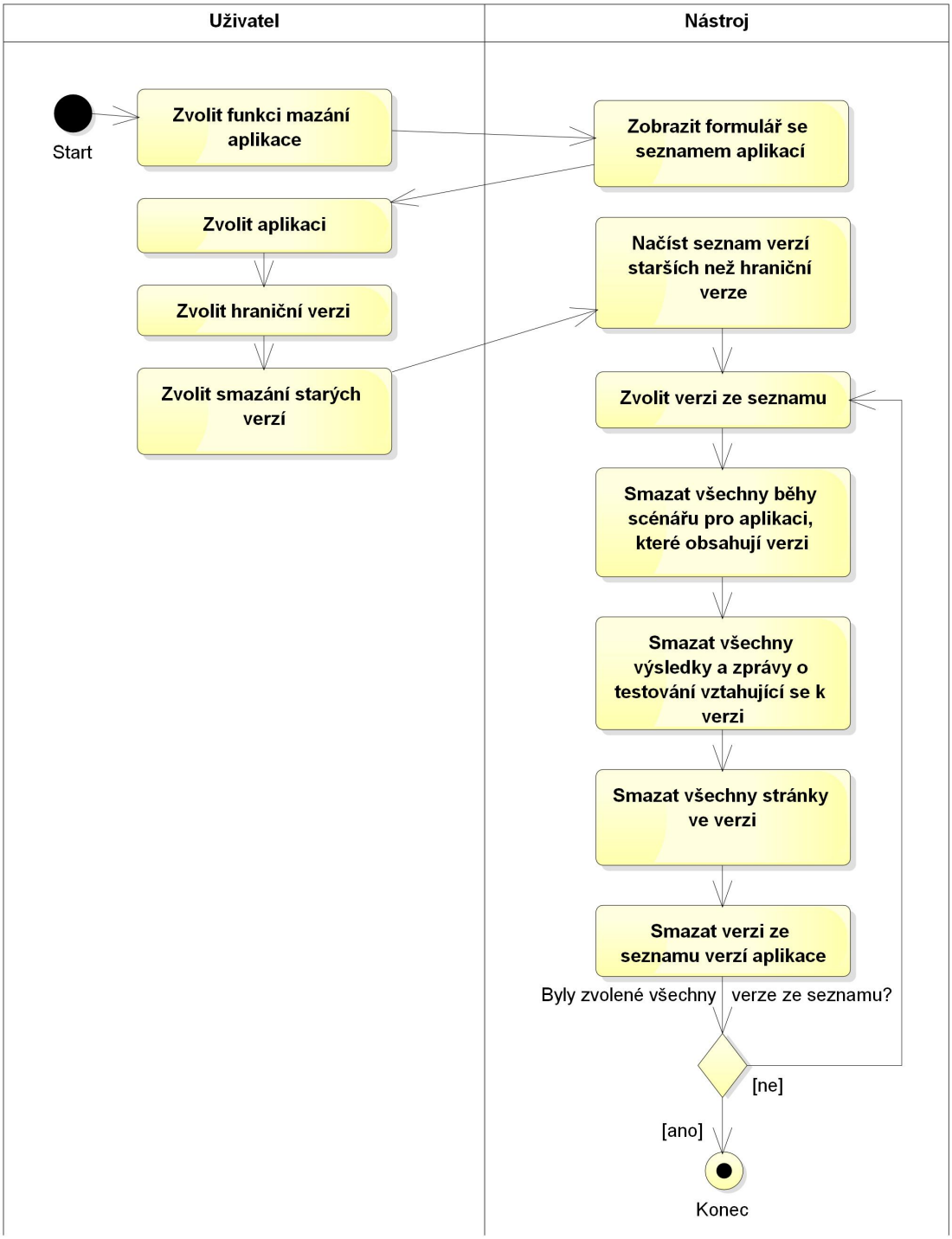
Diagram 2.8 popisuje proces pro smazání celé aplikace. Uživatel pouze vybere aplikaci, kterou chce smazat, a Nástroj ji smaže, musí však předtím odstranit všechny informace navázané na aplikaci.



Obrázek 2.8. Diagram aktivit pro smazání aplikace

2.5.9 Smazání starých verzí aplikace

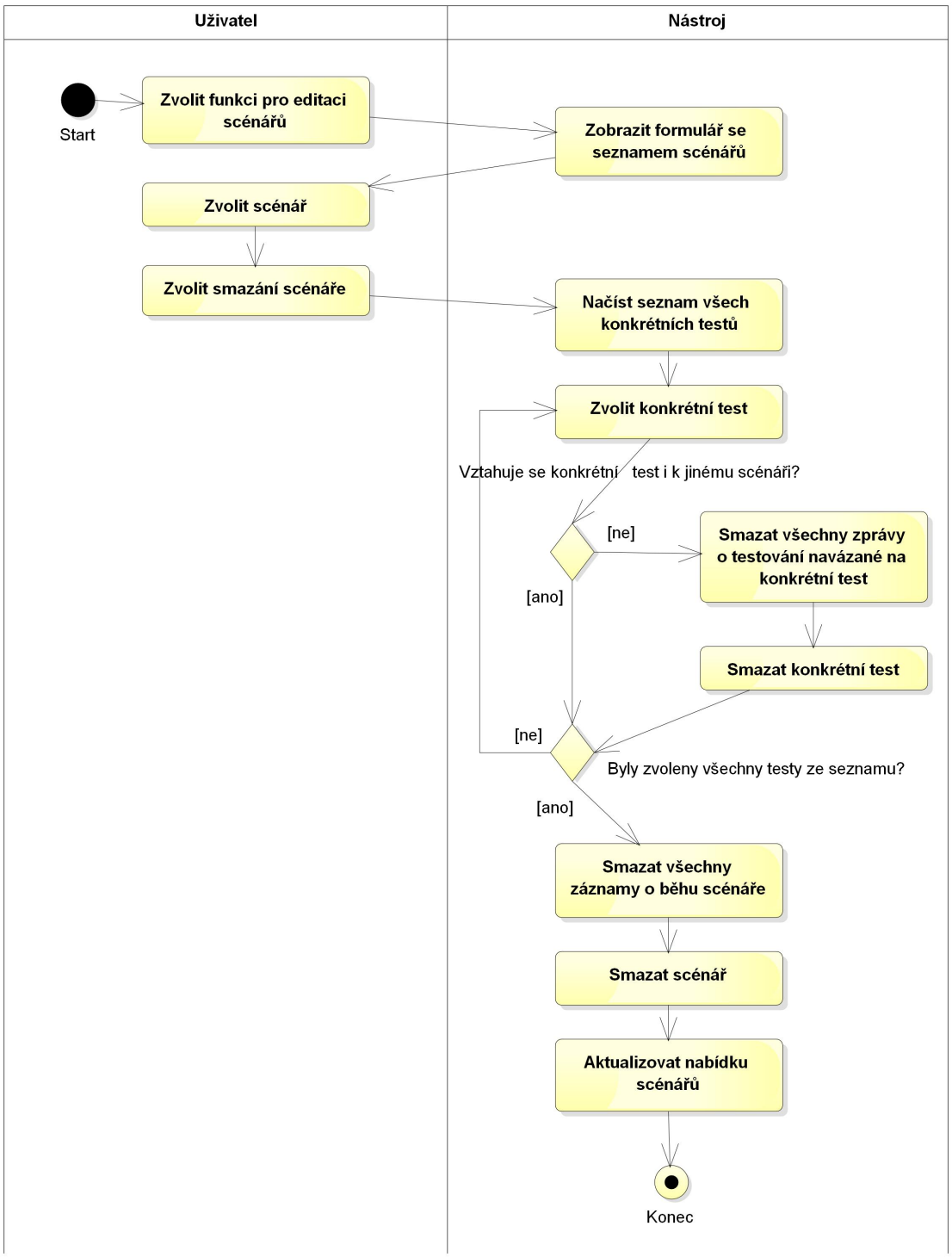
Diagram 2.9 popisuje proces smazání verzí aplikace, které jsou starší než zvolená verze. Uživatel zvolí aplikaci a z jejích verzí vybere tu hraniční, kterou chce ještě ponechat. Všechny verze, které jsou staršího data, než je zvolená hraniční verze, budou smazány. Nástroj smaže všechny stránky aplikace stažené ve starých verzích a všechny informace o testování, které jsou na staré verze navázané.



Obrázek 2.9. Diagram aktivit pro smazání starých verzí aplikace

2.5.10 Smazání testovacího scénáře

Diagram 2.10 zachycuje proces mazání testovacího scénáře. Pokud uživatel nechce nadále scénář používat a nezajímají ho výsledky konkrétních testů, které do scénáře patřily, může scénář smazat. Tato funkce se liší od funkce zneaktivnění scénáře, která scénář pouze skryje v nabídce scénářů pro další testování.



Obrázek 2.10. Diagram aktivit pro smazání testovacího scénáře

2.6 Seznam testovacích technik a jejich definice

Nyní popíši jednotlivé techniky, pomocí kterých lze srovnávat dvě verze stejné stránky, případně rovnou porovnat všechny stránky v aplikaci. Popíši zde, jak budou techniky fungovat pro jednotlivé stránky a co budou hodnotit, bude-li srovnávána celá aplikace. V případě testování celé aplikace navíc každá technika spočítá, kolik stránek z první verze chybí oproti druhé a naopak.

2.6.1 Test na počet všech elementů

Tato technika spočítá, kolik elementů (HTML tagů) je v kódu stránky a zjistí rozdíl v počtu elementů u dvou časových verzí stránky. Technika také zjistí, kolik elementů mělo v obou verzích stejný obsah.

Pro celou aplikaci technika zjistí, kolik se vyskytuje elementů na všech stránkách dohromady a kolik elementů celkem má stejný obsah v obou verzích.

2.6.2 Test na počet výskytů zadaného podřetězce

Test vyhledá všechny výskyty zadaného podřetězce v kódu stránky. Jako podřetězec může být zadán i regulární výraz. Je možné vyhledávat přesně podle velikosti písmen (case sensitive), nebo nezávisle na velikosti (case insensitive). Technika poté srovná výskyt podřetězce v obou verzích.

Pro celou aplikaci technika srovná celkový výskyt podřetězce v obou verzích.

2.6.3 Test na počet výskytů zadaného elementu

Test vyhledá, kolikrát se v kódu vyskytuje element s daným názvem (tagem) a porovná výskyt tohoto elementu v obou verzích stránky nebo aplikace.

2.6.4 Test na počet výskytů zadaného atributu

Test vyhledá, kolikrát se v kódu vyskytuje atribut s daným názvem a porovná výskyt tohoto atributu v obou verzích stránky nebo aplikace.

2.6.5 Test na stabilitu atributu

Parametrem testu je název atributu. Test vyhledá všechny atributy s takovým názvem, porovná tyto atributy v obou verzích a rozřadí je do čtyř skupin:

- XPath atributu a jeho hodnota jsou stejné v obou verzích
- XPath atributu je stejná v obou verzích, ale hodnota se liší
- XPath atributu není stejná v obou verzích, ale hodnota atributu je stejná
- XPath atributu i hodnota atributu jsou v obou verzích odlišné

Některé atributy se mohou vyskytovat zároveň ve skupině 2 a 3, protože mohou mít stejnou hodnotu, jako nějaký atribut, který leží na jiné XPath a zároveň mohou mít stejnou XPath jako atribut s jinou hodnotou.

Pro celou aplikaci sečte test počet atributů v těchto čtyřech skupinách přes všechny stránky.

2.6.6 Test na počet validačních chyb a varování

Test zjistí počet validačních chyb a počet varování pro stránku. Poté zhodnotí, jestli stránka je nebo není validní. Validitu zhodnotí podle kritéria, kolik chyb a kolik varování je možné tolerovat. Toto kritérium určuje uživatel.

Pro celou aplikaci zjistí test celkovou sumu validačních chyb a varování a spočítá také procentuální podíl validních a nevalidních stránek v aplikaci.

2.6.7 Test na rozdílná a stejná místa v kódu

Test srovná zdrojový kód stránky ve dvou časových verzí znak po znaku a vyhledá podřetězce, ve kterých se stránky liší a ve kterých se shodují. Výstupem testu bude kód s graficky vyznačenými odlišnými místy (pro každou verzi bude jiná poznávací značka), počet míst, na kolika se stránky liší a pro každou verzi počet znaků, v kolika se kód liší oproti srovnávané verzi. Technika spočítá i procentuální poměr změněných znaků.

Pokud stránka v jedné z verzí neexistuje, znamená to pro existující verzi, že se změnilo 1 místo (celá stránka) a 100% znaků (všechny znaky).

Pro celou aplikaci test spočítá počet míst, na kolika se aplikace liší, počet změněných znaků celkem, průměrná procenta změněných znaků. Navíc sečte, kolik stránek se změnilo a kolik procent z počtu všech stránek v aplikaci toto číslo představuje.

2.6.8 Test na výskyt vložených prvků

Možnými prvky, které komplikují automatické testování, jsou např. AJAXové elementy, vložené aplikace Flash, Active X, stránky přesměrovávané JavaScriptem a další [1]. Tento test bude mít seznam regulárních výrazů, pomocí kterých tyto prvky v kódu identifikuje a spočítá. Seznam výrazů identifikujících vložené prvky si bude moct uživatel upravit podle svých potřeb.

Výstupem testu bude počet výskytů jednotlivých výrazů ze seznamu, pro celou aplikaci to pak bude součet výskytů těchto výrazů přes všechny stránky.

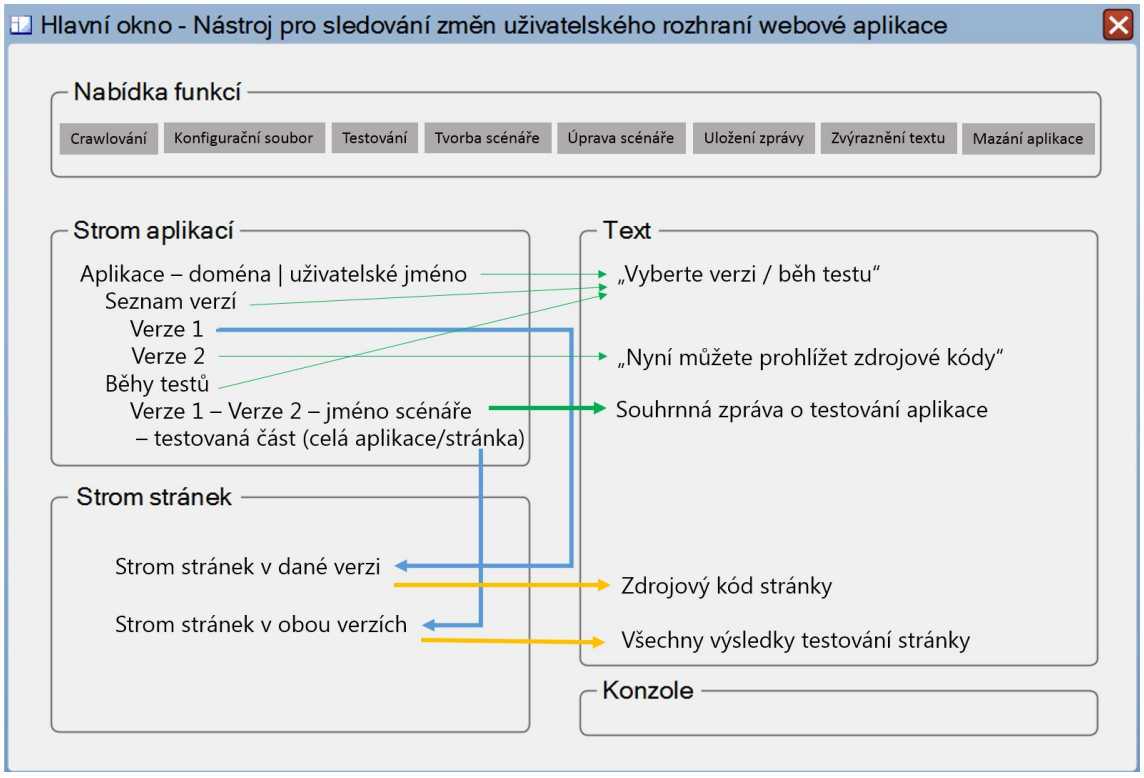
2.7 Návrh uživatelského rozhraní

Hlavní okno uživatelského rozhraní bude nabízet přehled funkcí, bude zobrazovat strom aplikací a strom stránek. Jak jsem výše definovala, uživatel by měl mít možnost vracet se k výsledkům testů a zároveň by měl mít možnost prohlížet zdrojové kódy stránky. Rozhodla jsem se proto zobrazit strom aplikací a podle volby uživatele se bude měnit strom stránek a zpráva v textové části hlavního okna. Tyto změny jsou zachyceny v návrhu 2.11 šipkami, které znázorňují, co se zobrazí při výběru aplikace či stránky ze stromu.

Text v hlavním okně je měněn i testovací funkcí, která zobrazuje souhrnný výsledek pro proběhlé testování.

Pod textem se nachází konzole, která upozorňuje na průběh crawlování, varuje uživatele a také jej informuje o chybách, které se mohou stát při práci s neočekávanými daty.

Při volbě funkce z nabídky se otevře vždy nové okno pro funkci. Vyvolání funkce bude možné stisknutím tlačítka. Tlačítko bude z estetických důvodů mít ikonu pro funkci a stručný popis funkce.



Obrázek 2.11. Návrh hlavního okna uživatelského rozhraní

Každá funkce zobrazí po zvolení nové okno, v němž bude formulář pro údaje potřebné k vyplnění funkce. Tato okna budou otevírána pod sebou nalevo od hlavního okna, aby mohl uživatel mezi okny snadno přepínat. Bude se jedna o malá okna, pouze s formulářem. Jedinou výjimkou je zde funkce „Zvýraznění textu,“ která může otevřít velké okno se zvýrazněním textem z hlavního okna.

Funkce *Konfigurační soubor*, *Testování*, *Tvorba scénáře*, *Úprava scénáře*, *Uložení zprávy* a *Mazání aplikace* zobrazí vždy jen jedno okno, bez ohledu na to, kolikrát jsou funkce zavolány. Bude-li už okno funkce otevřeno, přenese se při dalším zavolání funkce

na vrch. Okna těchto funkcí budou zobrazovány pod sebou v pořadí, v jakém byly nyní vyjmenovány. Uživatel tak bude mít přístup k přepínání mezi otevřenými okny.

Funkce *Crawlování* a *Zvýraznění textu* budou otevírat nové okno při každém zavolání.

Všechna okna budou přemístitelná.

Nyní popíši návrhy pro okna jednotlivých funkcí.

Crawlování

Tato funkce zahrnuje ukládání stránek k analýze, tedy jak crawlování aplikace, tak stahování sady stránek. Název funkce *Crawlování* je vybrán kvůli potřebě krátkého textu v nabídce funkcí.

Tato funkce otevře vždy nové okno, aby mohl uživatel spustit více crawlování najednou a sledovat přitom běh všech crawlovacích robotů souběžně. Ve stejném okně bude možné stahovat i sadu stránek – uživatel jen vybere konfigurační soubor a spustí crawlování nebo stahování a aplikační logika Nástroje se postará o zbytek. Okna se budou otevírat pod sebou. Pokud uživatel některá okna zavře, další nová okna přeberou polohu dříve zavřených oken. První otevřené okno se zobrazí pod okny ostatních funkcí, které mají svou polohu předem danou.

Okno bude obsahovat prostor pro zprávy uživateli, nabídku konfiguračních souborů a tlačítko pro spuštění crawlování. Když bude spuštěno crawlování aplikace, místo tlačítka na spuštění se objeví tlačítka na pozastavení či zastavení procesu. Při pozastavení se objeví místo tlačítka na pozastavení tlačítko pro pokračování procesu. Když je proces ukončen, zobrazí se opět původní nabídka pro nové crawlování.

Konfigurační soubor

Okno funkce bude mít tři karty – kartu pro vytvoření konfiguračního souboru pro crawlování, kartu pro vytvoření konfiguračního souboru pro stahování sady stránek a kartu pro mazání konfiguračních souborů. Název funkce *Konfigurační soubor* je vybrán kvůli potřebě krátkého textu v nabídce funkcí.

Karta pro vytvoření konfiguračního souboru pro crawlování bude obsahovat formulář s následujícími položkami: Oblast pro zprávy Nástroje uživateli, nabídka pro výběr souboru, do nějž bude konfigurační soubor uložen (jako filepicker), pole pro výchozí URL, možnost zaškrtnout tvorbu konfiguračního souboru pro přihlášené crawlování (po zaškrtnutí přibydou do formuláře vstupy pro informace k přihlašování) a tlačítko pro vytvoření konfiguračního souboru.

Karta pro vytvoření konfiguračního souboru pro stahování sady stránek bude obsahovat formulář s následujícími položkami: Oblast pro zprávy Nástroje uživateli, tlačítko pro vytvoření konfiguračního souboru, tlačítko pro přidání další stránky a pole pro URL stránek (počet těchto polí je neomezen, tlačítko pro přidání další stránky bude přidávat další pole, ve formuláři bude možné rolovat.)

Karta pro mazání konfiguračních souborů bude obsahovat nabídku konfiguračních souborů a tlačítko, jež smaže zvolený konfigurační soubor.

Testování

Tato funkce zahrnuje analýzu uložené stránky či aplikace pomocí definovaného scénáře. Název funkce *Testování* je vybrán kvůli potřebě krátkého textu v nabídce funkcí.

Okno bude reagovat na zvolenou aplikaci nebo stránku v hlavním menu. Pokud je v hlavním menu zvolená nějaká aplikace, okno nabídne otestování této aplikace. Pokud bude zvolená nějaká stránka, okno nabídne otestování jen této stránky, ale uživatel

bude moci zaškrtnout, že si přeje otestovat celou aplikaci. Pokud v hlavním okně není nic zvoleného, požádá okno testování nejprve o zvolení.

Okno bude mít pouze jeden formulář, který bude obsahovat tyto prvky: Oblast pro zprávy Nástroje uživateli, volbu dvou verzí (pokud je v hlavním okně nějaká verze zvolena, nabídne ji okno testování jako první), nabídku testovacích scénářů a tlačítko pro spuštění testování. Okno bude při testování průběžně informovat uživatele o tom, kolik stránek aplikace již bylo otestováno. Po skončení testování se okno zavře a výsledky se zobrazí v hlavním okně.

Tvorba scénáře

Obsah okna pro tvorbu scénáře se bude měnit podle fáze, vždy bude ale obsahovat oblast pro zprávy Nástroje uživateli. V první fázi bude mít okno formulář pro volbu jména scénáře. Ve druhé fázi nabídne okno seznam testovacích technik, uživatel vybere jednu z nich a přejde do třetí fáze. Ve třetí fázi se zobrazí vstupy pro vyplnění parametrů dané techniky. Vyplněním parametrů a pokračováním přidá uživatel do scénáře konkrétní test. Pokud technika žádné parametry nepotřebuje, přeskočí okno do čtvrté fáze, kde se může uživatel rozhodnout o přidání dalšího konkrétního testu do scénáře, nebo o vytvoření scénáře. Pokud se uživatel rozhodne pro přidání dalšího konkrétního testu, objeví se opět formulář z druhé fáze. Pokud se uživatel rozhodne pro vytvoření scénáře, Nástroj scénář vytvoří a v okně se zobrazí opět první fáze a informace o vytvoření scénáře.

Úprava scénáře

Tato funkce zahrnuje zobrazení scénářů, jejich úpravu a mazání. Název funkce *Úprava scénáře* je vybrán kvůli potřebě krátkého textu v nabídce funkcí.

Toto okno bude mít formulář, který bude obsahovat oblast pro zprávy Nástroje uživateli a nabídku scénářů. Poté, co uživatel zvolí scénář, přidají se k formuláři následující prvky:

- Informace o tom, zda je scénář aktivní či neaktivní a tlačítko, které může tuto vlastnost změnit.
- Tlačítko pro smazání scénáře.
- Přehled konkrétních testů. Lze zvolit některý z těchto testů a posunout ho ve scénáři o pozici výš nebo níž. Také lze konkrétní test ze scénáře odstranit.
- Tlačítko pro přidání nového konkrétního testu do scénáře.

Kdykoli lze zvolit jiný test, poté se změní druhá část formuláře.

Uložení zprávy

Zvolí-li uživatel funkci pro uložení zprávy, zobrazí se v okně pro ukládání zpráv vždy nová karta. Tato karta bude obsahovat formulář s oblastí pro zprávy Nástroje uživateli, ve které bude uživatel informován, jakou zprávu ukládá. Dále bude formulář požadovat zvolení souboru, do nějž bude zpráva uložena a bude obsahovat tlačítko pro uložení zprávy. Když Nástroj zprávu uloží, zavře i kartu, která se zprávy týkala.

Zvýraznění textu

Tato funkce otevře vždy nové okno, aby mohl uživatel porovnávat více výstupů hlavního okna (např. si tak může uživatel otevřít vedle sebe výsledky testování stránky a zdrojové kódy v porovnávaných verzích). Toto okno formátuje výstup z hlavního okna: Pokud byl zobrazený zdrojový kód, zvýrazní syntax HTML kódu. Pokud byly zobrazeny výsledky testu, podtrhá jména testovacích technik a zvýrazní porovnávané verze.

Speciální funkci bude toto okno plnit při zobrazení výsledků testovací techniky Test na rozdílná a stejná místa v kódu, když rozdílná místa, která budou označena nějakou textovou značkou, nahradí tím, že rozdílný text obarví dvěma různými barvami.

Mazání aplikace

Okno této funkce bude nabízet seznam aplikací, tlačítko pro smazání celé aplikace a nabídku verzí, jež se bude pojit k tlačítku pro smazání verzích starších, než ta verze, která bude zvolená v nabídce.

Kapitola 3
Implementace

V této kapitole se budu zabývat implementací Nástroje. Uvedu použité technologie a ujasním použití knihoven třetích stran. Popíši řešení datového úložiště, které je rozděleno na databázi a souborový systém. Uvedu architekturu Nástroje, popíši implementaci testovacích technik a další části implementace.

3.1 Volba technologií

Pro Nástroj jsem zvolila formu desktopové aplikace. Nástroj tak bude moci ukládat stažené zdrojové kódy přímo do souborového systému v uživatelově počítači a zároveň držet informace o stažených stránkách v databázi na uživatelově počítači.

Nástroj jsem vytvářela v programovacím jazyku Java, pro vývoj jsem použila Java Standard Edition, verze 1.7. Kvůli komplexnosti projektu jsem se rozhodla použít formu projektu Maven, která je vhodná pro podnikové aplikace.

Pro různorodost funkcí implementovaného Nástroje jsem se rozhodla použít několik knihoven třetích stran, které obsahovaly funkce, jež bych jinak musela zdlouhavě implementovat. Tyto knihovny bych nyní ráda krátce popsala a uvedla jejich použití v mém Nástroji.

3.1.1 Log4j

Log4j [4] je open source API od Apache Software, které pomáhá zaznamenávat zprávy v průběhu aplikace. Uživatel tohoto API si může zvolit způsob, jakým budou zprávy zaznamenávány (např. zápis do souboru, výpis do konzole ad.) v konfiguračním souboru (XML, JSON, YAML, nebo properties soubor), nebo může konfiguraci naprogramovat. Log4j se skládá ze tří komponent: Logger (entita, kterou Apache servlet využívá na logování zpráv), Appender (jednotlivé způsoby záznamu zpráv) a Layout (určuje výsledný vzhled zpráv).

Nástroj pracuje často s instancí třídy org.apache.log4j.Logger, pomocí které zapisuje informace o běhu do souboru logging.log a do konzole. Logger umožňuje určit druh zprávy podle úrovně: DEBUG, INFO, WARN, ERROR a FATAL. Díky tomu se uživatel může rychle zorientovat ve zprávách, které od loggeru dostává.

3.1.2 HtmlUnit

HtmlUnit [5] plní funkci webového prohlížeče pro Javu, pouze nemá uživatelské rozhraní. Knihovna umí mimo jiné zpracovat HTML, vyplňovat formuláře, následovat odkazy a pracovat s XPath elementů na stránce. Autorem knihovny je Mike Bowler z Gargoyle Software, Id skupiny je net.sourceforge.htmlunit.

Důvod užití HtmlUnit v Nástroji je dvojí. Zaprvé s touto knihovnou pracuje použitý crawler. Zadruhé byla tato knihovna nejvhodnější pro testovací techniku na stabilitu hodnoty atributu, protože umožňuje efektní práci s XPath elementů.

3.1.3 Jsoup

Jsoup [6] je HTML parser pro Javu, který umí zpracovat i nevalidní HTML. Knihovna provádí rozbor HTML pomocí DOM, CSS a jquery-like metod, díky tomu je schopná určit jednotlivé složky HTML kódu.

Proto je Jsoup vhodným nástrojem pro různé testovací techniky, v nichž se počítají a porovnávají elementy a atributy.

3.1.4 JUnit

JUnit [7] je framework pro opakované jednotkové testování. Usnadňuje a koordinuje psaní automatických testů, kterými jsem v Nástroji ověřila funkcionalitu testovacích technik a základního běhu Nástroje včetně propojení s databází.

3.1.5 RSyntaxTextArea

RSyntaxTextArea [8] je editor pro aplikace užívají Java Swing (knihovna pro formulářové aplikace). Tento editor zvýrazňuje syntax kódu pro více než 40 programovacích jazyků. V Nástroji je použit proto, aby mohl uživatel zobrazit zformátovaný kód stažených HTML stránek.

3.1.6 Spring Data JPA

Framework Spring [9] je určen pro vývoj standardních i podnikových aplikací. Spring nabízí několik modulů s různorodou funkčností – např. propojení aplikací a webu, propojení databází a integrace dat, aspektově orientované programování, testování, model-view-controller, vzdálený přístup a další. [10]

Modul Spring Data umožňuje propojení javovské podnikové aplikace s datovými úložišti (relační i nerelační databáze, map-reduce Framework, datové servisy pro cloudová úložiště). [11]

Spring Data JPA dovoluje snadno implementovat JPA úložiště a přístup k datům. JPA znamená Java Persistence API – nástroj pro umožnění objektově-relačního mapování (ORM). Data, která jsou v programovacím jazyce zastoupena objekty, jsou konvertována do relační databáze, kde jsou data reprezentována tabulkami místo objektů.

V Nástroji slouží Spring Data JPA jako zprostředkovatel mezi aplikací a technologií Hibernate a pomáhá přenášet data z databáze a do databáze.

3.1.7 Hibernate

Hibernate je Framework, který umožňuje objektově-relační mapování. Framework mapuje javovské objekty na entity (tabulky) relační databáze a zachovává perzistenci těchto objektů. To znamená, že udržuje objekty stálé a odolné proti nechtěným změnám a stará se o to, aby aplikace pracovala s objekty se stejnými vlastnostmi, s jakými jsou uloženy v databázi. A stejně tak se stará o to, aby se změny, které aplikace provede s objekty, úspěšně uložily do databáze. [12]

Mapování může probíhat pomocí zápisu metadat do XML souborů, nebo pomocí anotací ve zdrojovém kódu. [13]

V Nástroji jsem se rozhodla použít anotace. Každá entita je zastoupena třídou. Tato třída je tzv. POJO – Plain Old Java Object, klasický objekt. Atributy entit jsou připojeny pomocí metod get/set. Vztahy s ostatními entitami jsou ve třídách znázorněny anotací (@ManyToOne, @OneToMany, @ManyToMany), kolekce (např. listy), které elementy vlastní, jsou propojeny anotací @ElementCollection. Zatímco v kódu jsou tyto kolekce vlastnictvím třídy, v relační databázi jsou kolekce zastoupeny tabulkami. Stejně tak jsou v databázi znázorněny dalšími tabulkami vztahy mezi entitami.

Hibernate se stará o spolupráci Nástroje a databáze.

3.1.8 PostgreSQL JDBC driver

PostgreSQL [14] je objektově-relační databáze. PostgreSQL JDBC driver (ovladač) je softwarový nástroj, který umožňuje spojení Java aplikací a PostgreSQL databáze. JDBC znamená Java Database Connectivity a jedná se o API, které definuje jednotné rozhraní pro přístup k relačním databázím. [15]

V Nástroji je driver použit pro spojení s databází PostrgreSQL, kterou jsem se rozhodla použít pro ukládání dat. V případě, že by se budoucí uživatelé rozhodli pro používání jiné databáze, bylo by nutné tento driver nahradit driverem pro databázi jiného, uživatelem preferovaného druhu.

3.1.9 Jcabi

Jcabi [16] je sbírka mnoha malých komponent pro obohacení aplikací v jazyce Java o různé funkce. V Nástroji je použita knihovna w3c , která slouží jako adaptér pro online W3C validátor. Navíc vyžaduje i závislost na frameworku Jersey [17], který se stará o propojení s REST (Representational State Transfer) webovými služby. Jersey vytváří webového klienta, jehož prostřednictvím knihovna Jcabi komunikuje s W3C validátorem.

Tyto knihovny jsou v Nástroji využity v testovací technice na validitu stránky podle validátoru W3C.

3.2 Převzatý modul - crawler

Pro procházení webových aplikací jsem potřebovala modul, který by automaticky navštívil všechny stránky aplikace a stáhl a uložil jejich HTML kód, představující uživatelské rozhraní. Požadavkům na toto procházení odpovídá crawler, který je implementován v rámci bakalářské práce *Nástroj pro hledání zadaných vzorů v uživatelském rozhraní webové aplikace* [3], ze které jsem čerpala informace o funkcionalitě crawleru.

Tento crawler jsem převzala. Do Nástroje ovšem není přidán jako knihovna, na které by byl Nástroj závislý, ale jsou přidány přímo zdrojové kódy. Je to tak z toho důvodu, že jsem crawler potřebovala mírně upravit, aby odpovídal požadavkům Nástroje. Upravený kód se nachází v balíčku cz.cvut.fel.crawl v podsložkách crawler a interfaces.

3.3 Řešení datového úložiště

Nástroj pracuje s velkým množství dat – ukládá mnoho zdrojových kódů v různých časech, konfigurační soubory pro crawlování, nabídku testovacích technik, testovací scénáře, výsledky analýzy různých stránek apod. Rozhodla jsem se zkombinovat ukládání do databáze a do souborového systému. Zdrojové kódy je pro jejich velikost lepší ukládat do souborů, navrhla jsem tedy způsob ukládání do složek. Informace o stažených aplikacích a o testech jsou zase příliš propojené a často kusé, pro jejich ukládání je tedy vhodné využít databázi. Realizaci obou úložišť nyní popíši.

3.3.1 Souborový systém

Aplikace používá souborový systém v uživatelově počítači ve třech případech:

1. **Ukládání stažených kódů**
   Nástroj ukládá všechny kódy do složky pro stažené aplikace. Zdrojové kódy jsou ukládány do textového souboru s koncovkou .html a jsou uloženy jako text, nikoli jako stromové HTML, protože stažený kód nemusí být validní.

   V hlavní složce pro ukládání stažených aplikací vytvoří Nástroj podsložku pojmenovanou jménem aplikace a uživatelským jménem. V této složce je pak vytvořena struktura podle webové stránky, jednotlivé verze jsou pojmenovány podle data a času stažení.

   Struktura vypadá například takto:

   ```
   C:\...\DownloadedPages\fel.cvut.cz anonymous\fel.cvut.cz\
   2016-04-24-11-52.html
   ```

2. **Ukládání výsledků analýzy stránky**
   Uživatel si může přímo z Nástroje uložit text zobrazený v hlavním okně uživatelského rozhraní, např. výsledky testovacího scénáře. Nástroj nabízí pro ukládání funkci, díky které uživatel uloží zobrazený text do textového souboru. Uživatel si může zvolit složku, do které chce soubor uložit, ale jako první mu bude nabídnuta složka podle nastavení.

3. **Načítání konfiguračních souborů pro stažení webové stránky**
   Uživatel může v Nástroji pomocí grafického rozhraní vytvořit konfigurační soubory pro stahování webové aplikace nebo sady stránek. Tyto konfigurační soubory jsou ukládány do jedné konkrétní složky, odkud jsou poté i načítány. Konfigurační soubory pro crawlování celé aplikace ze zadaného odkazu jsou ukládány do xml souborů. Konfigurační soubory pro stahování sady stránek jsou ukládány do souborů .txt, každá stahovaná stránka je uložena na nový řádek.

Uživatel si předem zvolí tyto tři cílové složky v konfiguračním souboru pro nastavení Nástroje, setting.xml. Umístění složky může uživatel kdykoli změnit, změna se projeví při příštím spuštění Nástroje. Především pro složku s konfiguračními soubory a pro složku se staženými kódy je nutné, aby uživatel překopíroval její obsah do nové lokace (Nástroj by jinak nemohl najít zdrojové kódy dříve stažených aplikací a dříve vytvořené konfigurační soubory).

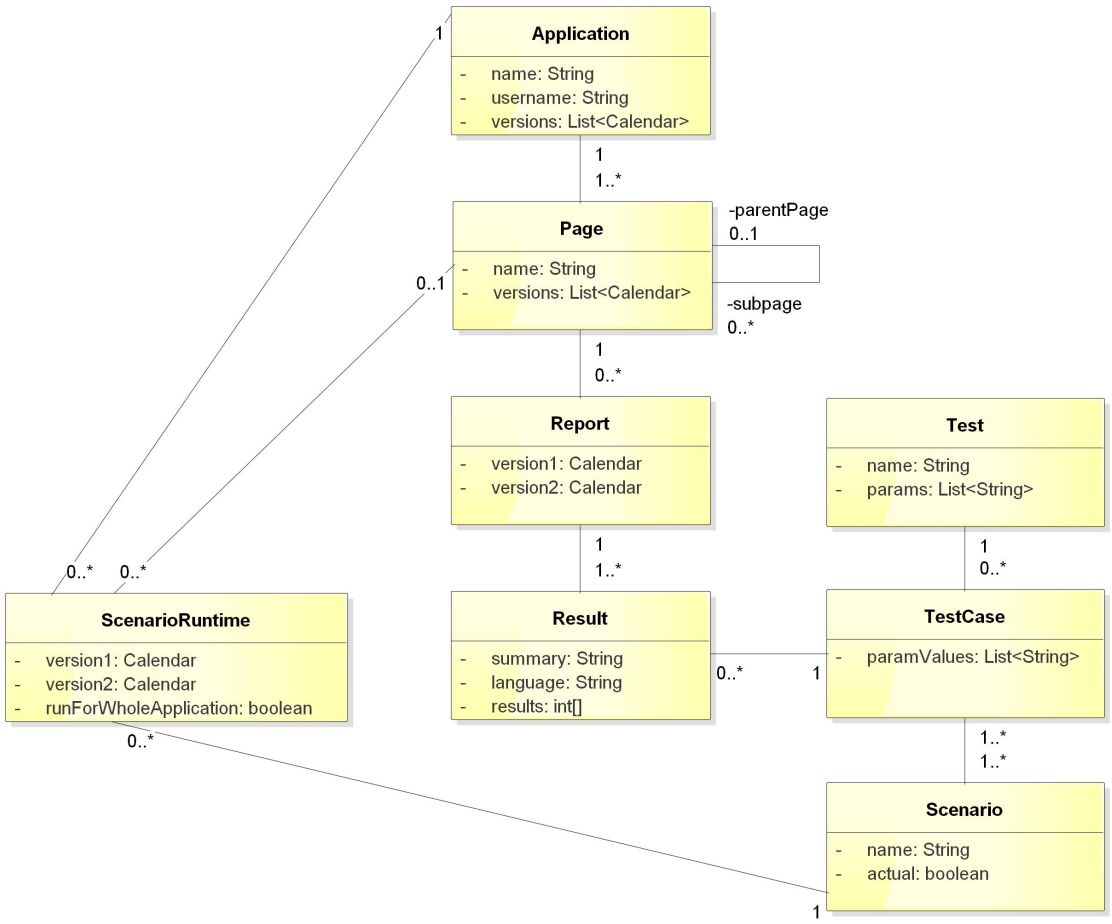
V případě, že by Nástroj nemohl nalézt nebo přečíst konfigurační soubor pro nastavení Nástroje, budou jako výchozí nastaveny složky s implicitním název a tyto složky budou uloženy v oblasti pro ukládání dat aplikací dle operačního systému, v němž je Nástroj spuštěn.

3.3.2 Databáze

Informace o stažených aplikacích, verzích crawlování a stromu stránek jsem se rozhodla uložit do databáze, stejně tak jako nabídku testovacích technik, testovací scénáře a údaje o jednotlivých běžích testovacích scénářů. Nástroj databázi využívá rovněž pro ukládání výsledků testů, čehož se využívá hlavně pro rychlé zobrazení výsledků již proběhlých testů a pro vyhodnocení analýzy pro celou aplikaci v okamžiku, kdy byly postupně testovány jednotlivé stránky, ne nutně všechny najednou.

Pro propojenost informací jsem se rozhodla navrhnout strukturu databáze tak, jak ji zachycuje schéma 3.1. Tento model je popsán v anglickém jazyce, protože byl podkladem pro databázi, která je psaná v angličtině, stejně jako zdrojový kód Nástroje. Proto nyní uvedu překlad do terminologie této práce.

Třída *Application* zastupuje aplikaci, třída *Page* stránku. Na tyto se váže *Scenario-Runtime* – běh scénáře. Scénář *Scenario* se skládá z několika *TestCase* – našich konkrétních testů. Ty se odkazují ke třídě *Test*, která zastupuje naše definované testovací techniky. Výsledky konkrétních testů jsou uloženy ve třídě *Result*, která se pojí se třídou *Report*, jež udržuje informace o tom, v jakých dvou verzích byla aplikace porovnávána.



Obrázek 3.1. Databázové schéma

Zvolila jsem pro práci databázi postgres pro uživatelskou přívětivost administrační konzole.

Pro propojení databáze s Nástrojem bylo potřeba pečlivě definovat přístup k databázi a propojení pomocí vlastního definovaného *DAO* (Data Access Object).

Pro připojení desktopové aplikace k databázi bylo potřeba udělat následující kroky:

- Přidání závislostí na ovladač pro PostgreSQL, na Spring framework a na Hibernate do souboru pom.xml v kompatibilních verzích
- Přidání souboru persistence.xml do balíčku META-INF. V tomto souboru je nastavení cest k databázi, nastavení přístupových informací a nastavení driveru
- Zažádání o správce entit

```
EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("entityManagerFactory");
EntityManager = entityManagerFactory.createEntityManager();
```

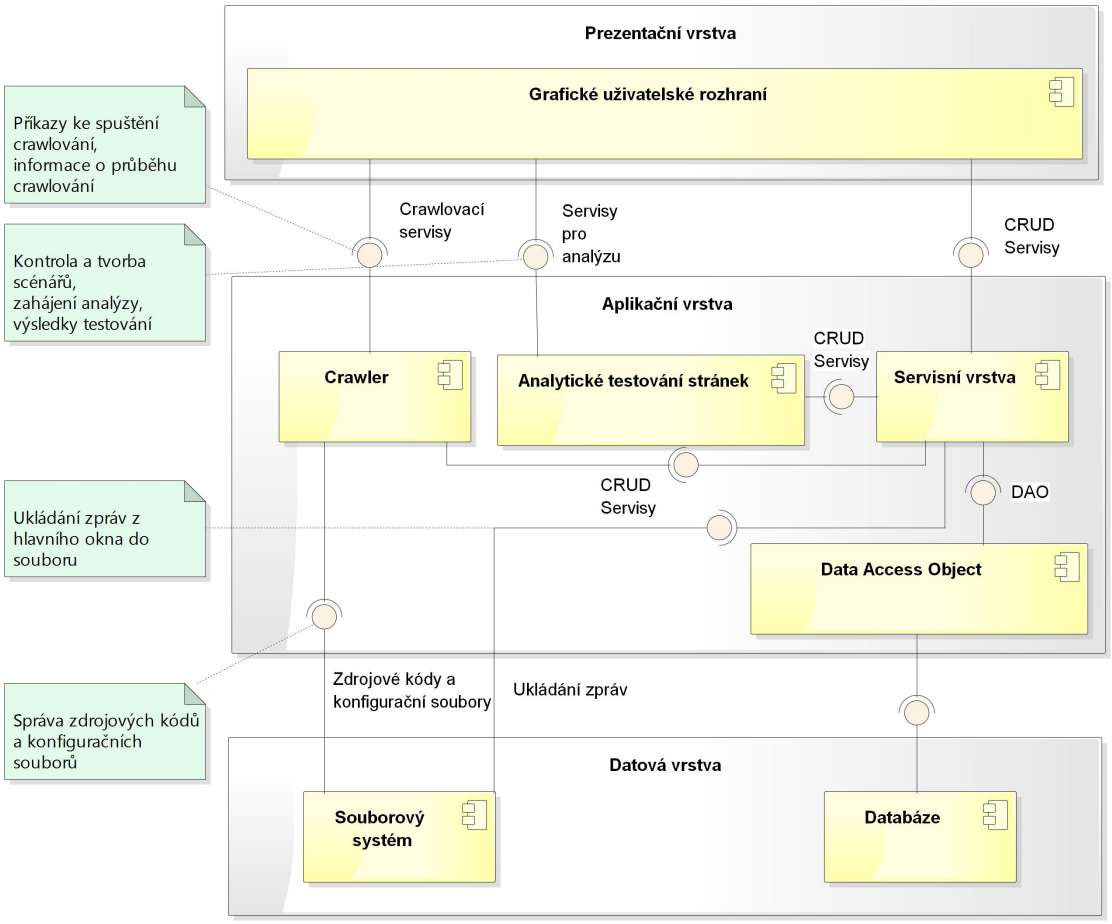
- Následné otevírání a provádění transakcí, např. transakce pro uložení entity:

```
public <T extends AbstractEntity> T save(T entity) {
   try {
      entityManager.getTransaction().begin();
      entityManager.persist(entity);
      entityManager.getTransaction().commit();
   } catch (Exception e) {
      entityManager.getTransaction().rollback();
   }
return entity;
}
```

- DAO objekty spravují ukládání, mazání a vyhledávání v databázi. Ke třídám objektů, které zastupují jednotlivé tabulky entit, přistupují pouze servisní třídy. Pro práci s entitami v jiných oblastech kódu používám zástupné objekty, ne však ve stejné struktuře, jaké mají entity, ale upravené pro potřeby metod, které s nimi pracují. Např. část aplikace, která se stará o testovací techniky, potřebuje přistupovat k entitám týkajících se této funkcionality. Nestává se tak přes tradiční DTO (Data Transfer Object), ale vytvořila jsem pro to zástupný objekt ObjectTest, ze kterého si servisní metody zjistí, co potřebují. Stejně tak vyhodnocuje testovací část Nástroje výsledky analýzy pomocí objektu TestResult.
- Poté bylo nutné vytvořit tabulky v databázi a uložit do databáze jména a parametry testovacích technik, viz. 3.5.

3.4 Architektura řešení

Nyní popíši architekturu řešení a rozdělení funkcionalit do jednotlivých modulů. Uvedu graf jednotlivých komponent a poté vysvětlím, o co se která stará. Komponenty lze rozdělit do jednotlivých vrstev.



Obrázek 3.2. Architektura řešení

Architekturu lze rozdělit na tři vrstvy – **prezentační**, která popisuje uživatelské rozhraní, **aplikační**, která se stará o výpočty a komunikaci mezi programy a o hlavní funkce aplikace, a **datovou**, která zahrnuje obě datová úložiště – souborový systém a databázi.

Prvky z vyšších vrstev přistupují k prvkům z vrstvy bezprostředně nižší.

Každé komponentě odpovídá v kódu jeden balíček. Mimo balíček je pouze hlavní třída a třída pro jazykovou lokalizaci. Hlavní třída se stará o spuštění Nástroje a o udržuje informace, které jsou nastavitelné uživatelem – tj. zvolené složky pro ukládání a volba jazyka Nástroje.

3.4.1 Prezentační vrstva

Prezentační vrstva se stará o uživatelské rozhraní, které je napsané pomocí knihoven Swing. Úkolem uživatelského rozhraní je zobrazení hlavního okna, zobrazování dalších oken podle zvolené funkce, zachycení uživatelské interakce s Nástrojem a základní kontrola uživatelských vstupů (zastavení odeslání nevyplněných políček formuláře apod.). Složitější kontrolu uživatelských vstupů (např. správně vyplněné parametry pro kon-

krétní test) a zpracování pokynů přenechává prezentační vrstva vrstvě aplikační, uživatele informuje o výsledku.

Prezentační vrstva je zastoupena balíčkem *GUI*. Pro jednotlivé funkce je v balíčku vždy jedna třída. V hlavním okně se zobrazuje seznam aplikací a stránek, komunikuje proto se servisní vrstvou, stejně jako funkce pro úpravu scénáře a pro mazání aplikace.

Se servisní vrstvou komunikuje též funkce pro testování, která interaguje i s komponentou pro testování. Za pomoci komponenty pro testování se také vytváří nový testovací scénář.

Funkce pro vytváření konfiguračního souboru a pro crawlování posílá pokyny a údaje do komponenty crawleru.

Funkce pro ukládání zpráv přebírá zprávu od hlavního okna a ukládá ji pomocí speciální třídy v servisní vrstvě do souboru.

Funkce pro zobrazení zvýrazněného výstupu otevře zprávu z hlavního okna vždy v novém okně. Nejen že zobrazí přehledně výstup, ale navíc umožňuje uživateli otevřít si více zpráv v nových oknech a vzájemně je porovnávat.

3.4.2 Aplikační vrstva

Nyní popíši jednotlivé komponenty, které se starají o aplikační logiku Nástroje.

Crawler - tato komponenta se stará o crawlování aplikace a stahování sady stránek a o vše, co k tomu náleží. V kódu je komponenta zastoupena balíčkem *crawl*, který má kromě tříd v kořenové složce, které jsou součástí vlastní implementace Nástroje, také dvě podsložky – crawler a interfaces. V těchto podsložkách je mírně upravený kód crawleru převzatého ze zdroje [3].

V kořenové složce jsou třídy, které se starají o vytváření a mazání konfiguračních souborů, o spuštění a tok crawlování, o stahování sady stránek a o ukládání a načítání zdrojových kódů.

Analytické testování stránek – tato komponenta se stará o zásadní funkci Nástroje – zjišťuje změny uživatelského rozhraní webové aplikace. Komponenta je zastoupena balíčkem *analysis*, v podbalíčku *test* jsou implementovány testovací techniky. Komponenta se stará o vytvoření scénáře, kontroluje uživatelem vyplněné parametry při vytváření konkrétních testů a hlavně spouští analýzu. Analýzu lze spustit pro jednotlivou stránku či pro celou aplikaci. Třída Analyser umí podle názvu konkrétního testu určit, jaká testovací technika má být na stránku či aplikaci zavolána.

Servisní vrstva – tato komponenta, zastoupena balíčkem *service*, se stará především o ukládání, zobrazování, upravování a mazání dat z databáze. Tyto úpravy provádí pomocí Data Access Objectu. Vrstva má pro práci s databází tři servisy: servis pro správu aplikací (stará se o aplikace a jejich stránky, správu jednotlivých verzí, načítá stromy pro uživatelské rozhraní), servis pro správu testování (stará se o testovací scénáře, zaznamenává a vyhledává zprávy o testování apod.) a servis pro mazání (zajišťuje mazání aplikace a mazání scénářů). Navíc má servis pro ukládání zpráv do souborů.

Data Access Object, nebo také DAO, česky Objekt pro přístup k datům, je komponenta, která může manipulovat s daty v databázi. Komponenta je zastoupena balíčkem *dao*, který obsahuje rozhraní s metodami, které má DAO splňovat. Rozhraní využívá javovská generika – objekt je schopen manipulovat s generickým typem T. T musí být však stejného typu, jako abstraktní entita. Toto řešení umožňuje univerzální manipu-

laci se všemi entitami databáze a redukuje tak kód pro práci s databází. Komponenta obsahuje i implementaci DAO rozhraní a třídu, jež tvoří manažera entit, který spravuje jednotlivé tabulky (entity) v databázi.

3.4.3 Datová vrstva

Datovou vrstvou je míněno datové úložiště, které je popsáno v kapitole 3.3. Datové úložiště je rozděleno na dvě části – jednotlivé složky, které jsou v kódu zastoupeny cestou ke složce, a databáze, jejíž tabulky jsou namapované na entity, které jsou popsány v balíčku model.

3.5 Implementace testovacích technik

Každá testovací technika musí mít dvě metody – jednou porovná jednu stránku ve dvou verzích, druhou porovná dvě verze celé aplikace. V obou případech musí metoda dodat dvojí výsledek:

- text, který popisuje hodnoty zjištěné testovací technikou a shrne je uživateli stručnou zprávou,
- pole se spočítanými hodnotami.

Text slouží jako výstup pro uživatele, informuje ho o tom, co potřebuje o testovaných stránkách či aplikacích vědět. Pole spočítaných hodnot je důležité pro logiku Nástroje – pokud se analyzuje celá aplikace, vyhodnocuje testovací technika souhrnné výsledky pomocí výsledků pro jednotlivé stránky. Každá testovací technika má svá vlastní pravidla, jak má pole spočítaných hodnot vypadat, kolik hodnot má obsahovat a jak tyto informace využít pro výpočet souhrnných výsledku pro aplikaci.

Nyní popíši důležité části implementace jednotlivých technik.

3.5.1 Test na počet všech elementů

Parametry: žádné

Počet elementů zjišťuji pomocí knihovny Jsoup [6] a její metody getAllElements().

3.5.2 Test na počet výskytů zadaného podřetězce

Parametry: podřetězec, citlivost na velikost písmen

Výskyt zadaného podřetězce zjišťuji pomocí knihovny java.util.regex, kde je zadaný podřetězec použit jako vyhledávaný regulární výraz. Podle přání uživatele jsou výrazy shodné se zadáním hledány s ohledem či bez ohledu na velikost písmen výrazu (tzv. case sensitive nebo case insensitive).

3.5.3 Test na počet výskytů zadaného elementu

Parametry: jméno elementu

Počet elementů se zadaným jménem (tagem) zjišťuji pomocí knihovny Jsoup [6] a její metody getElementsByTag(String tagName).

3.5.4 Test na počet výskytů zadaného atributu

Parametry: jméno atributu

Test vyhledá všechny elementy s daným atributem pomocí knihovny Jsoup [6] a její metody getElementsByAttribute(String attributeName). Počet takových elementů odpovídá počtu atributů, protože atribut stejného jména nesmí být přiřazen stejnému elementu víckrát než jednou [18].

3.5.5 Test na stabilitu atributu

Parametry: jméno atributu

Při vyhodnocování tohoto testu využívám knihovnu HtmlUnit [5], pro kterou jsem se rozhodla, protože umí (na rozdíl od Jsoup) vyhodnotit XPath k hledaným HTML elementům. Knihovna HtmlUnit nejprve načte pomocí metody WebClient.getPage(String path) zdrojový kód do objektu HtmlPage, ze kterého získá metodou getByXpath(String xpathExpr) seznam elementů, jejichž XPath obsahuje zadaný výraz, kterým je v našem případě výraz `//*[@jménoAtributu]`.

Test si zapamatuje seznam všech elementů se zadaným jménem a uloží si jejich hodnoty a XPath. Poté vytřídí atributy, které jsou pro obě stránky stejné. Následně zjistí atributy, které mají stejnou hodnotu v obou stránkách, ale jinou XPath a vybere i atributy, které mají stejnou XPath, ale jinou hodnotu. Atributy, které dosud nebyly přiřazeny k žádné skupině, jsou atributy, které se ve srovnávané verzi stránky vůbec nevyskytují.

3.5.6 Test na počet validačních chyb a varování

Parametry: počet tolerovaných chyb, počet tolerovaných varování

Test využívá knihovnu com.jcabi.w3c [16], pomocí které odešle kód stránky na validaci validátorem W3C, který nalezne všechny validační chyby a varování podle standardů konsorcia W3C. V implementaci nevyužívám přímo validátor této knihovny, protože obsahoval chybu, kterou nebylo možné pouze za používání knihovny obejít – validátor občas nebyl schopný přečíst nějaký znak z odpovědi W3C validátoru a kvůli tomu nebyl schopný pracovat s odpovědí dál. Místo toho využívám vlastní validátor, který ale vychází z knihovny jcabi a používá některé součásti knihovny, které chybné nejsou.

Test pracuje se zjištěnými počty validačních chyb a varování, které srovná s uživatelem zadaným počtem tolerovaných chyb a varování. Pokud bylo nalezeno méně nedostatků, než jsou tyto limity, označí test stránku za validní.

3.5.7 Test na rozdílná a stejná místa v kódu

Parametry: žádné

Základní myšlenka implementace byla, aby výsledek testu vypadal podobně jako při použití nástroje google-diff-match-patch [19]. Na základě tohoto požadavku jsem implementovala metodu, která srovnává dva textové řetězce pomocí algoritmu Eugene W. Myerse [20], který nejlépe a nejrychleji vyhodnocuje rozdíly v kódu. Bohužel, úskalí tohoto algoritmu spočívá v jeho prostorové náročnosti, kdy se alokuje dvourozměrné pole, jehož dimenze jsou velké jako délky porovnávaných textových řetězců. Pro srovnávání velkých řetězců je tedy algoritmus nevhodný, jelikož by mohlo docházet k přetékání haldy.

Z toho důvodu jsem se rozhodla zkombinovat Myersův algoritmus s algoritmem pro vyhledávání nejdelšího společného podřetězce. Z obou kódů vyextrahuji nejdelší společný podřetězec a to opakuji tak dlouho, dokud nejsou řetězce dostatečně malé na to, aby se mohl použít Myersův algoritmus bez rizika přetečení haldy. Nejdelší společný podřetězec vyhledávám pomocí algoritmu, jehož prostorová náročnost je pouze O(n), cenou za to je ale vysoká časová náročnost, která je O(n2).

3.5.8 Test na výskyt vložených prvků

Parametry: žádné

Tato testovací technika má v souboru *embeddedObjects.xml*, který se nachází v kořenové složce projektu, seznam regulárních výrazů, pod kterými by se měly skrývat

vložené prvky. Test prozkoumá zdrojové kódy a spočítá pro každou položku seznamu zvlášť počet výskytů regulárního výrazu. Výraz se vyhodnocuje bez ohledu na velikost písmen.

Seznam si může uživatel sám upravit tak, aby byly identifikovány takové prvky, které ho zajímají, protože je dnes již mnoho způsobů a mnoho druhů aplikací, které jde do stránek vložit a nelze předem odhadnout, které budou uživatele zajímat. Úprava se projeví až v nově spuštěných testech.

V případě, že by se soubor se seznamem nenacházel v kořenové složce projektu, použije Nástroj implicitní seznam, který je uložený ve vnitřní struktuře Nástroje.

Do implicitního souboru bylo implementováno několik vzorových identifikátorů. Mezi ně patří:

- identifikátor pro vyhledávání vloženého elementu applet (nyní již zastaralý způsob), který spouští externí Java aplikace. [21]

```
<applet code="jmeno.class" ... >
```

- identifikátory pro zástupce aplikace vložené pomocí objektu:

```
<OBJECT ... CODETYPE="application/druh-aplikace" ... >
```

  Jako zástupce je uveden identifikátor pro vložené flashové aplikace - application/x-shockwave-flash, pro vložené java aplikace - application/java a pro aplikace Microsoft Word - application/msword
- identifikátor vloženého souboru takového typu, který není běžně podporován, ale pomocí plug-inů se zobrazí [22]. Jako zástupce je uveden identifikátor pro videa přehrávaná pomocí přehrávače VLC [23].

```
<embed type="application/x-vlc-plugin"
pluginspage="http://www.videolan.org"/>
```

- identifikátor vložených skriptů, které by mohly vykonávat předpokládanou činnost. Jako zástupce je uveden skript, v němž probíhá zavolání ajaxovací funkce z JQuery [24]:

```
 <script> ... $.ajax( ... ) ... </script>
```

3.6 Další části implementace

3.6.1 Jazyková lokalizace

Nástroj je internacionalizován pro český a výchozí anglický jazyk. Uživatelské rozhraní i zprávy o testování se zobrazují ve zvoleném jazyce. Pokud byla stránka nebo aplikace testována pouze v jazyce, který právě není zvolen, je uživateli oznámeno, že stránka byla testována, ale ne pro právě zvolený jazyk. Uživatel musí testy zavolat znovu pro současný jazyk.

Pokud chce uživatel změnit jazyk, musí tak učinit před spuštěním Nástroje. Pokud již Nástroj běží, nereaguje na změnu jazyka v nastavení, protože tuto informaci načítá při spouštění.

3.6.2 Vzájemná interakce částí uživatelského rozhraní

Protože Nástroj spravuje několik oken, je nutné informace v otevřených oknech průběžně měnit, když jsou data ovlivněna jinými funkcemi. V Nástroji dochází k několika interakcím mezi jednotlivými okny:

Aktualizace stromu aplikací v hlavním okně – ke změnám dochází, je-li provedené nové crawlování či stahování sady stránek, testování, nebo když je nějaká aplikace smazána. Ke změně dochází také tehdy, je-li přejmenován nějaký scénář (kvůli větvi běhů scénářů), nebo když je nějaký scénář odstraněn.

Aktualizace stromu stránek v hlavním okně – ke změně dochází vždy, když uživatel vybere jinou aplikaci ve stromu aplikací. K aktualizaci dochází také při změně stromu aplikací.

Aktualizace okna pro testování – k této dochází tehdy, byla-li provedena nějaká změna v seznamu scénářů.

Aktualizace okna pro crawlování – k této dochází tehdy, byla-li provedena nějaká změna v seznamu konfiguračních souborů.

Aktualizace záložky pro mazání konfiguračních souborů – k této dochází tehdy, byl-li vytvořen nový konfigurační soubor, nebo když byl nějaký konfigurační soubor smazán.

3.6.3 Kontrola nechtěného zavření okna

Pokud chce uživatel zavřít hlavní okno Nástroje, ujistí se Nástroj, že nejde o omyl. Neúmyslným vypnutím by totiž uživatel mohl přijít o výsledky testování, mohl by zastavit crawlování aplikace, které již nelze v daném čase opakovat a mohl by ztratit právě rozpracovaný nový scénář. Pokud tedy uživatel bude chtít zavřít hlavní okno a poběží některý z těchto procesů, Nástroj ho upozorní. Pokud i přesto uživatel Nástroj zavře, testování se zruší, crawlování zastaví a scénář se neuloží.

Uživatel je upozorněn i při vypínání okna crawlování, které právě probíhá, při vypínání okna s rozpracovaným scénářem a při vypínání okna s probíhajícím testováním.

Kapitola 4
Testování

Vyvinutý Nástroj bylo nutné pečlivě otestovat. Testování bylo rozděleno na tři úrovně testování:

Vývojářské testy – Kód byl kontrolován několika jednotkovými testy, které se nacházejí v balíku *Test Packages*. Detaily těchto testů jsou uvedeny v kapitole 4.1. Dále byla funkcionalita Nástroje otestována několika manuálními testy, jejich scénáře jsou uvedeny v kapitole 4.2. Dle výsledků jednotkových a manuálních testů jsem opravila drobné chyby v kódu a provedla testování znovu. Proces vývojářského testování tedy ověřil správnost hlavních funkcí aplikace.

Integrační testy, systémové testy – Nástroj, sestavený do spustitelného souboru jar, byl ozkoušen s Javou verze 1.7 a s databází Postgres 9.4. Za těchto podmínek byl Nástroj ozkoušen na zástupcích jednotlivých operačních systémů. Nástroj byl úspěšně spuštěn v OS Windows 7, v OS iMac late 2013 a v OS Linux Fedora 23.

Uživatelské akceptační testy – Uživatelské akceptační testy byly realizovány v rámci pravidelných konzultací s vedoucím bakalářské práce a Nástroj byl nezávisle ověřen dvěma uživateli (studentem softwarového inženýrství a programátorkou obchodních aplikací). Vedle volného procházení aplikace byly použity testy uvedené v kapitole 4.2.

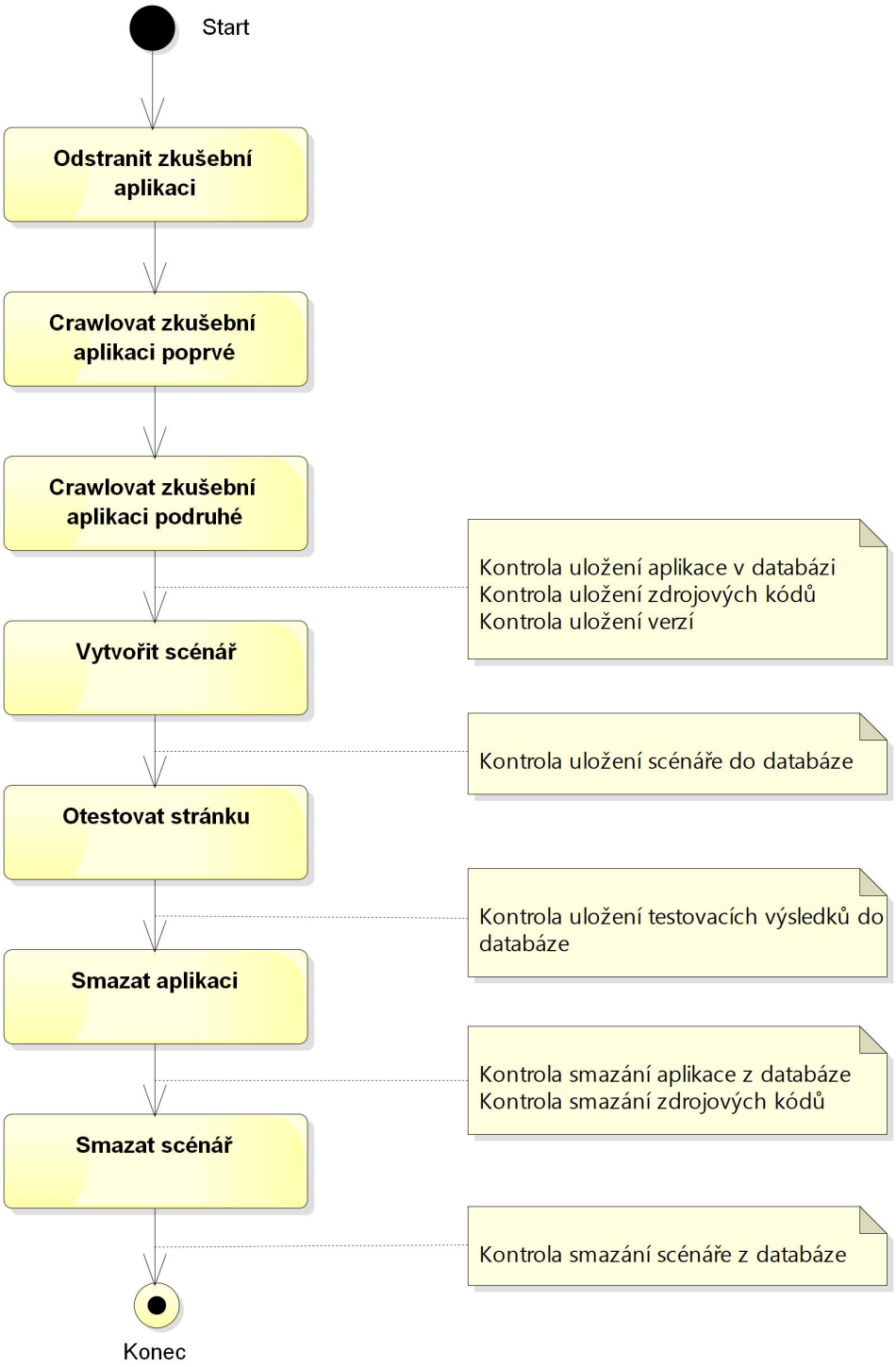
4.1 Jednotkové testy

Pro testování kódu bylo napsáno několik jednotkových testů.

Pro každou testovací techniku byl napsán JUnit test, který kontroluje správnost techniky pro testování jedné stránky a pro testování celé aplikace. Pro kontrolu správnosti testování jedné stránky byly uměle vytvořeny zdrojové kódy, které ověřují různé aspekty techniky. Pro testování celé aplikace jsou technice zadány výsledky testování imaginárních stránek a JUnit test kontroluje, zda technika z těchto výsledků spočítá správný souhrnný výsledek.

Další JUnit test kontroluje, zda jsou v databázi uloženy názvy a parametry všech testovacích technik, aby Nástroj správně fungoval.

Poslední JUnit test se zaměřuje na správnost hlavního toku aplikace a na funkčnost základních funkcí servisní vrstvy. Pro tento test byl předpřipraven konfigurační soubor pro crawlování aplikace. V databázi by neměla být uložena aplikace „file“ s uživatelským jménem „anonymous,“ protože bude v průběhu JUnit testu smazána. Průběh JUnit testu je zachycen na diagramu 4.1.



Obrázek 4.1. Průběh testu hlavního toku

4.2 Manuální testy

Pro interakci uživatele s Nástrojem a pro funkce Nástroje, které nebyly otestovány JUnit testy, bylo provedeno manuální testování. Postup manuálního testování a jednotlivé kroky ověřování funkčnosti Nástroje lze shrnout do scénářů pro manuální testování. Nyní popíši scénáře základních manuálních testů.

ID	1
Testovaná funkce	Vytvoření konfiguračního souboru
Vstupní podmínky	Žádné
Testovací kroky a očekávané výsledky	1. Zvolte funkci „Konfigurační soubor.“ 2. Po zobrazení okna přejděte do karty „Tvorba konfiguračního souboru pro crawlování aplikace“ 3. Stiskněte tlačítko „Vyberte soubor nebo jméno souboru“. Až se vám zobrazí formulář, napište do pole pro jméno souboru „forum“. Potvrďte. 4. Zadejte do pole „Startovní URL“ URL nějakého diskuzního fóra, kam je nutné se přihlašovat. 5. Zvolte, že chcete crawlovat jako přihlášený uživatel. 6. Ověřte, že ve formuláři přibyly vstupy pro přihlášené crawlování. 7. Vyplňte správně vstupy pro přihlášené crawlování. Zadejte své přístupové údaje pro přihlášení do diskuzního fóra a XPath cestu k elementům, kam se budou tyto údaje na výchozí URL stránce vyplňovat. 8. Stiskněte tlačítko „Vytvořit nový konfigurační soubor pro crawlování“ 9. Ověřte, že byl ve Vámi zvolené složce pro ukládání konfiguračních souborů vytvořen soubor forum.xml. Ověřte, že byly do souboru vloženy všechny Vámi vyplněné vstupy.

ID	2
Testovaná funkce	Vytvoření konfiguračního souboru pro stahování sady stránek
Vstupní podmínky	Žádné
Testovací kroky a očekávané výsledky	1. Zvolte funkci „Konfigurační soubor.“ 2. Po zobrazení okna přejděte do karty „Tvorba konfiguračního souboru pro stahování sady stránek“ 3. Stiskněte tlačítko „Vyberte soubor nebo jméno souboru“. Až se vám zobrazí formulář, napište do pole pro jméno souboru „sada“. Potvrďte. 4. Klikněte 3x na tlačítko „Přidat další stránku“. Do políček pro URL stránek zadejte tyto hodnoty: „Neexistující stránka,“ „https://www.seznam.cz/,“ „http://mapy.cz,“ a „pocasi.seznam.cz/“. Jedno pole nechte prázdné. 5. Stiskněte tlačítko „Vytvořit konfigurační soubor pro stažení sady stránek“ 6. Ověřte, že byl ve Vámi zvolené složce pro ukládání konfiguračních souborů vytvořen soubor sada.txt. Ověřte, že byly do souboru vloženy všechny Vámi vyplněné vstupy a že nebylo uložené prázdné pole. Soubor by měl mít 4 řádky se zadanými adresami.

ID	3
Testovaná funkce	Přihlášené crawlování
Vstupní podmínky	Vytvořený konfigurační soubor „forum.xml“ ze scénáře č. 1
Testovací kroky a očekávané výsledky	1. Zvolte funkci „Crawlování.“ 2. Zvolte konfigurační soubor „forum.xml“ a stiskněte tlačítko „Crawluj!“. 3. Ověřte, že se zahájí stahování stránek, a že Nástroj neupozorní na nezdařené přihlášení. Ověřte, že se objeví tlačítko „Pozastavit“ a „Zastavit“. Ověřte, že Nástroj stahuje správné stránky a to i ty stránky, na které se dostanou jen přihlášení uživatelé. 4. Stiskněte tlačítko „Pozastavit“. Ověřte, že Nástroj přestal stahovat stránky (V oblasti pro informování uživatele se neobjevují nové zprávy o stažených stránkách). Ověřte, že se místo tlačítka „Pozastavit“ zobrazilo tlačítko „Pokračovat“. 5. Stiskněte tlačítko „Pokračovat“. Ověřte, že stahování stránek se opět spustilo. 6. Stiskněte tlačítko „Zastavit“. Ověřte, že se stahování zastavilo a že se opět objevila nabídka s konfiguračními soubory a tlačítko „Crawluj!“ 7. Ověřte, že Nástroj uložil zdrojové kódy do složek a uložil je pod správným datem. 8. Ověřte, že Nástroj uložil do databáze jméno aplikace a čas stahování jako novou verzi. Ověřte, že Nástroj uložil do databáze všechny stránky a jejich verzi. 9. Ověřte, že se aktualizoval strom stránek v hlavním okně a že přibyl strom pro stahovanou aplikaci diskuzního fóra. Rozbalte tento strom a ověřte, že ve větvi verzí je i čas, v němž jste zahájili crawlování.

ID	4
Testovaná funkce	Stažení sady stránek
Vstupní podmínky	Vytvořený konfigurační soubor „sada.txt“ ze scénáře č. 2, existující stránky „https://www.seznam.cz/“ a „http://mapy.cz“
Testovací kroky a očekávané výsledky	1. Zvolte funkci „Crawlování.“ 2. Zvolte konfigurační soubor „sada.txt“ a stiskněte tlačítko „Crawluj!“. 3. Ověřte, že se zahájí stahování stránek. Ověřte, že Nástroj nestáhne stránku „Neexistující stránka,“ úspěšně stáhne stránky „https://www.seznam.cz/“ a „http://mapy.cz“ bez ohledu na protokol a přitom nestáhne stránku „pocasi.seznam.cz/,“ protože protokol nebyl vyplněn.

ID	5
Testovaná funkce	Vytvoření scénáře
Vstupní podmínky	Neexistující scénář se jménem „Zkušební“
Testovací kroky a očekávané výsledky	1. Zvolte funkci „Tvorba scénáře.“ Pojmenujte scénář „Zkušební“ a stiskněte tlačítko „Pokračovat“. 2. Zvolte techniku „Počet elementů“ a stiskněte tlačítko „Přidat konkrétní test“ 3. Postupně přidejte do scénáře všechny testovací techniky. Ověřte, že Nástroj brání odeslání špatných uživatelských vstupů – zkoušejte přidat testy bez vyplněných parametrů, zkuste do polí, která vyžadují číselnou hodnotu zadat různé znaky, do polí, jež vyžadují specifický řetězec (např. „ano“ nebo „ne“) zadávat různé znaky. 4. Stiskněte tlačítko „Vytvořit.“ Ověřte, že do databáze byly uloženy všechny konkrétní testy a to bez duplicit, že byly uloženy všechny parametry tak, jak jste je zadali a že byl uložen scénář se všemi konkrétními testy. 5. Ověřte, že se opět zobrazil formulář pro nový scénář. Zadejte jméno „Zkušební.“ Ověřte, že Nástroj zabrání duplicitě a požádá Vás o jiné jméno.

ID	6
Testovaná funkce	Testování aplikace
Vstupní podmínky	Libovolná aplikace stažená alespoň ve dvou verzích, alespoň jeden testovací scénář
Testovací kroky a očekávané výsledky	1. Nezvolte v hlavním okně žádnou aplikaci. Zvolte funkci „Testování.“ Ověřte, že Vás Nástroj požádá o zvolení aplikace. 2. Zvolte v hlavním okně nějakou aplikaci a ve větvi verzí zvolte libovolnou verzi. Zvolte funkci „Testování.“ Ověřte, že se zobrazí okno s formulářem pro zvolenou aplikaci. Ověřte, že v poli pro verzi 1 je předvolená verze, kterou jste vybrali v hlavním okně. 3. Zvolte dvě stejné verze a nějaký scénář. Stiskněte tlačítko „Otestovat“. Ověřte, že Vás Nástroj požádá o volbu dvou odlišných verzí. 4. Zvolte dvě odlišné verze. Stiskněte tlačítko „Otestovat“. Ověřte, že Vás Nástroj informuje, kolik stránek z celkového počtu již otestoval. 5. Až testování skončí, ověřte, že se v hlavním okně zobrazil výsledek testování. Ověřte, že ve stromu aplikace je ve větvi „Běhy scénářů“ uloženo právě vykonané testování. 6. Klikněte na běh právě vykonaného testování. Ověřte, že se zobrazí výsledky testování scénářem pro aplikaci. 7. Zvolte libovolnou stránku ze stromu stránek. Ověřte, že se zobrazí výsledky všech testů, které pro tuto stránku proběhly.

ID	7
Testovaná funkce	Úprava scénáře
Vstupní podmínky	Vytvořený testovací scénář „Zkušební“ ze scénáře č. 5. Alespoň jedna aplikace testovaná tímto scénářem.
Testovací kroky a očekávané výsledky	1. Zvolte funkci „Úprava scénáře.“ Vyberte scénář „Zkušební“. Ověřte, že se ukáže formulář se správnými údaji a všemi konkrétními testy, které jste do scénáře přidali. 2. Ověřte, že je scénář aktivní. Zvolte funkci „Testování“ a ověřte, že je scénář v nabídce scénářů. 3. Stiskněte tlačítko „Označit jako neaktivní.“ Ověřte, že se tlačítko změnilo na „Označit jako aktivní.“ Zvolte funkci „Testování“ a ověřte, že scénář už není v nabídce scénářů. Ověřte, že informace o tomto scénáři nezmizely z větve „Běhy scénářů“ u aplikací, které byly tímto scénářem testovány. 4. Stiskněte tlačítko „Označit jako aktivní.“ Ověřte, že se tlačítko změnilo na „Označit jako neaktivní.“ Zvolte funkci „Testování“ a ověřte, že scénář je opět v nabídce scénářů. 5. Zvolte libovolný konkrétní test a posouvejte jím nahoru a dolu. Ověřte, že se jeho pozice mění. Ověřte, že pozici prvního konkrétního testů nelze posunout níž a že pozici posledního konkrétního testů nelze posunout výš. 6. Odeberte libovolný konkrétní test. Ověřte, že jsou testy ze scénáře odebírány, a že nelze smazat poslední konkrétní test ve scénáři. 7. Přidejte do scénáře nové konkrétní testy. Ověřte, že jsou testy přidány a ověřte, že nejsou přidány duplicitně. 8. Přejmenujte scénář. Ověřte, že scénář byl v nabídce přejmenován. Ověřte, že scénář byl přejmenován v bězích scénářů v hlavním okně ve stromě aplikací. 9. Smažte scénář. Ověřte, že scénář zmizel z nabídky scénářů. Ověřte, že běhy scénáře v hlavním okně ve stromě aplikací byly odstraněny. Ověřte, že byly odstraněny všechny konkrétní testy, které byly navázány pouze na smazaný scénář. Ověřte, že byly smazány výsledky těchto konkrétních testů.

ID	8
Testovaná funkce	Uložení zprávy
Vstupní podmínky	Žádné
Testovací kroky a očekávané výsledky	1. Zvolte funkci „Uložení zprávy.“ Ověřte, že se otevřelo okno pro ukládání zpráv s kartou, která obsahuje počátek vaší zprávy. 2. Nechte na hlavní stránku načíst jinou zprávu. Zvolte opět funkci „Uložení zprávy.“ Ověřte, že se v okně pro ukládání zpráv otevřela další karta, která obsahuje počátek druhé zprávy. 3. Vyberte soubor a uložte zprávu. Ověřte, že se do souboru skutečně uložila celá zpráva. Ověřte, že se karta zavřela. Opakujte i pro druhou zprávu.

ID	9
Testovaná funkce	Zvýraznění textu
Vstupní podmínky	Alespoň jedna stažená stránka ve dvou verzích
Testovací kroky a očekávané výsledky	1. Otevřete v hlavním okně zdrojový kód nějaké stránky. Zvolte funkci „Zvýraznění textu.“ Ověřte, že se zobrazilo nové okno se staženým zdrojovým kódem se zvýrazněnou HTML syntaxí. 2. Proveďte na nějakou stránku Test na rozdílná a stejná místa v kódu. Načtěte tyto výsledky do hlavního okna výběrem stránky ve stromě stránek. Zvolte funkci „Zvýraznění textu.“ Ověřte, že jsou rozdílná místa vyznačená barevně. Ověřte, že jsou jména testů zvýrazněna tučně. Ověřte, že jsou porovnávané verze zvýrazněné tučně a modrou barvou.

ID	10
Testovaná funkce	Mazání aplikace
Vstupní podmínky	Alespoň jedna stažená aplikace
Testovací kroky a očekávané výsledky	1. Zvolte některou zkušební staženou aplikaci. Zvolte funkci „Mazání aplikace.“ Zvolte funkci „Testování“ pro zkušební aplikaci. Smažte aplikaci. 2. Ověřte, že byla aplikace včetně stránek a všech výsledků zpráv odstraněna z databáze, ze složek a ze stromu aplikací v hlavním okně. Ověřte, že nyní nelze pro aplikaci provést testování.

4.3 Výsledky testování

Testování pomohlo odhalit několik drobných chyb, například Nástroj neodchytil výjimku v případě, když kontroloval startovní URL pro konfigurační soubor a stránka na URL se odkazovala na jiné stránky (např. bannery), Nástroj také špatně kontroloval vyplnění parametrů na citlivost na velikost písmen u testovací techniky na výskyt řetězce. Nalezené chyby byly opraveny a opravené části Nástroje byly otestovány znovu. Po opakovaném testování již nebyly nalezeny žádné další chyby.

Kapitola 5
Závěr

Cílem této bakalářské práce bylo vytvořit Nástroj pro sledování změn uživatelského rozhraní webové aplikace a proces vývoje zdokumentovat přes fázi analýzy a návrhu, implementace a testování realizovaného řešení. Nástroj byl implementovaný jako desktopová aplikace v programovacím jazyce Java. Vytvořený Nástroj bude pomáhat testerům webových aplikací v analýze testované aplikace a zefektivní tak jejich práci.

Nástroj umí stáhnout celou webovou aplikaci pomocí následování odkazů z úvodního URL. Stažené verze Nástroj porovnává osmi testovacími technikami, které sledují změny v elementech a atributech stránky, výskyty různých podřetězců a prvků, validitu stránky a rozdílná místa v kódu. Uživatel může vytvořit testovací scénáře a urychlit tak opakovaný proces analyzování aplikace.

Nástroj lze nadále rozšiřovat. Je možné přidat další testovací techniky, stejně jako upravovat techniky stávající. Část analýzy by mohla probíhat už za procesu stahování stránek (například analýza doby odpovědi stránek). Nástroj by mohl být obohacen o další funkce, které zefektivní analýzu, nebo by mohl být spojen s jinými nástroji, které se zabývají podobnou problematikou, například s *Nástrojem pro hledání zadaných vzorů v uživatelském rozhraní webové aplikace* [3].

Během vývoje a psaní bakalářské práce jsem získala mnoho nových zkušeností a zužitkovala jsem znalosti nabyté během mého bakalářského studia. Nejdůležitější pro mne bylo vyzkoušet si samostatnou implementaci a řízení většího projektu, během které jsem zdokonalila své programovací znalosti, snažila jsem se vylepšit styl a přehlednost mého programování a návrhu aplikace a také jsem získala lepší přehled o tom, jak dlouho mi vývoj trvá a kolik úprav a aktivit je potřeba i poté, co se zdá aplikace téměř naimplementovaná. Během realizace bakalářské práce jsem také prohloubila své poznatky z oblasti webových technologií a naučila jsem se pracovat s javovskými knihovnami, s nimiž jsem se dříve nesetkala.

Literatura

[1] BUREŠ, Miroslav. Metrics for Automated Testability of Web Applications. *Proceedings of the 16th International Conference on Computer Systems and Technologies*. ACM, 2015, 83-89.

[2] BUREŠ, Miroslav. Framework for assessment of web application automated testability. *Proceedings of the 2015 Conference on research in adaptive and convergent systems*. ACM, 2015, 512-514.

[3] KOULA, Ondřej. *Nástroj pro hledání zadaných vzorů v uživatelském rozhraní webové aplikace*. Praha, 2016. Bakalářská práce. České vysoké učení technické v Praze.

[4] *Log4j – Overview - Apache Log4j 2* [online]. 2015 [cit. 2016-04-28]. Dostupné z: https://logging.apache.org/log4j/2.x/manual/

[5] *HtmlUnit - Welcome to HtmlUnit* [online]. 2016 [cit. 2016-04-28]. Dostupné z: http://htmlunit.sourceforge.net/

[6] *Jsoup Java HTML Parser* [online]. 2016 [cit. 2016-04-28]. Dostupné z: https://jsoup.org/

[7] *JUnit - About* [online]. 2016 [cit. 2016-04-28]. Dostupné z: http://junit.org/junit4/

[8] *RSyntaxTextArea* [online]. 2015 [cit. 2016-04-28]. Dostupné z: https://bobbylight.github.io/RSyntaxTextArea/

[9] *Spring Data JPA* [online]. 2016 [cit. 2016-04-28]. Dostupné z: http://projects.spring.io/spring-data-jpa/

[10] *Guides (Spring)* [online]. 2016 [cit. 2016-04-28]. Dostupné z: https://spring.io/guides

[11] *Spring Data* [online]. 2016 [cit. 2016-04-28]. Dostupné z: http://projects.spring.io/spring-data/

[12] *Hibernate ORM* [online]. 2016 [cit. 2016-04-28]. Dostupné z: http://hibernate.org/orm/

[13] Object/relational mapping metadata. BAUER, Christian a Gavin KING. *Java persistence with Hibernate : revised edition of Hibernate in action.* 2. Rev. ed. Greenwich: Manning Publications Co., 2007, s. 123-140. ISBN 1-932394-88-5.

[14] *PostgreSQL JDBC About* [online]. 2015 [cit. 2016-04-28]. Dostupné z: https://jdbc.postgresql.org/about/about.html

[15] What is JDBC? HAMILTON, Graham, Cattell RICK a Maydene FISHER. *JDBC Database Access with Java : a tutorial and annotated reference.* 1. Boston: Addison-Wesley Publishing Company, 1997, s. 5. ISBN 0-201-30995-5.

[16] *Jcabi-w3c - W3C Java Validators* [online]. 2015 [cit. 2016-04-28]. Dostupné z: http://w3c.jcabi.com/

[17] *Jersey* [online]. 2016 [cit. 2016-04-28]. Dostupné z: https://jersey.java.net/

[18] *HTML syntax - HTML5: 4.4. Attributes* [online]. 2013 [cit. 2016-04-28]. Dostupné z: https://www.w3.org/TR/html-markup/syntax.html

[19] *Google-diff-match-patch* [online]. 2011 [cit. 2016-04-28]. Dostupné z: https://code.google.com/p/google-diff-match-patch/

[20] MYERS, Eugene. An O(ND) Difference Algorithm and its Variations. *Algorithmica.* 1986, 1(2), 251-266.

[21] Vkládáme aplety jazyka Java. CASTRO, Elizabeth. *HTML, XHTML a CSS : názorný průvodce tvorbou WWW stránek.* 1. Brno: Computer Press, 2007, s. 306. ISBN 978-80-251-1531-2.

[22] *Objekty v HTML: applet, object, embed* [online]. 2016 [cit. 2016-04-28]. Dostupné z: http://www.jakpsatweb.cz/html/objekty.html

[23] *Documentation:WebPlugin - VideoLAN Wiki* [online]. 2016 [cit. 2016-04-28]. Dostupné z: https://wiki.videolan.org/Documentation:WebPlugin/

[24] *JQuery.ajax() — jQuery API Documentation* [online]. 2016 [cit. 2016-04-28]. Dostupné z: https://api.jquery.com/jquery.ajax/

Příloha A
Návod k instalaci

Instalace *Nástroje pro sledování změn uživatelského rozhraní webové aplikace* (dále jen Nástroj) vyžaduje nejprve instalaci databáze a stažení Javy. Aplikace je ozkoušena pro Javu SE verze 1.7 a databázi Postgres 9.4.

Po úspěšné instalaci databáze a Javy můžete spustit uložený jar. Spustit ho můžete např. z terminálu následujícím příkazem:

```
java -jar cesta_k_souboru.jar
```

Databáze PostgreSQL a nástroj pgAdmin je ke stažení zde:
`http://www.postgresql.org/download/`
Javu je možné stáhnout zde:
`http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`

A.1 Vybrání jazyka a složek pro práci s aplikací

Budete-li spouštět Nástroj ze souboru jar, je dobré mít ve stejné složce jako tento soubor i soubory *setting.xml* a *embeddedObjects.xml* pro funkci testovací techniky na výskyt vložených prvků. Pokud tyto soubory nebudou ve stejné složce, použije Nástroj výchozí nastavení pro složky a jazyk a použije také výchozí soubor pro detekci appletu, který je uložen ve zdrojích Nástroje.

Před spuštěním může uživatel nastavit složky pro práci Nástroje v souboru *setting.xml*, aby Nástroj rovnou ukládal dokumenty tam, kam je uživatelem požadováno. Pokud tyto složky nebudou nastaveny, bude Nástroj ukládat a čerpat data ze složky pro aplikace předem určené operačním systémem.

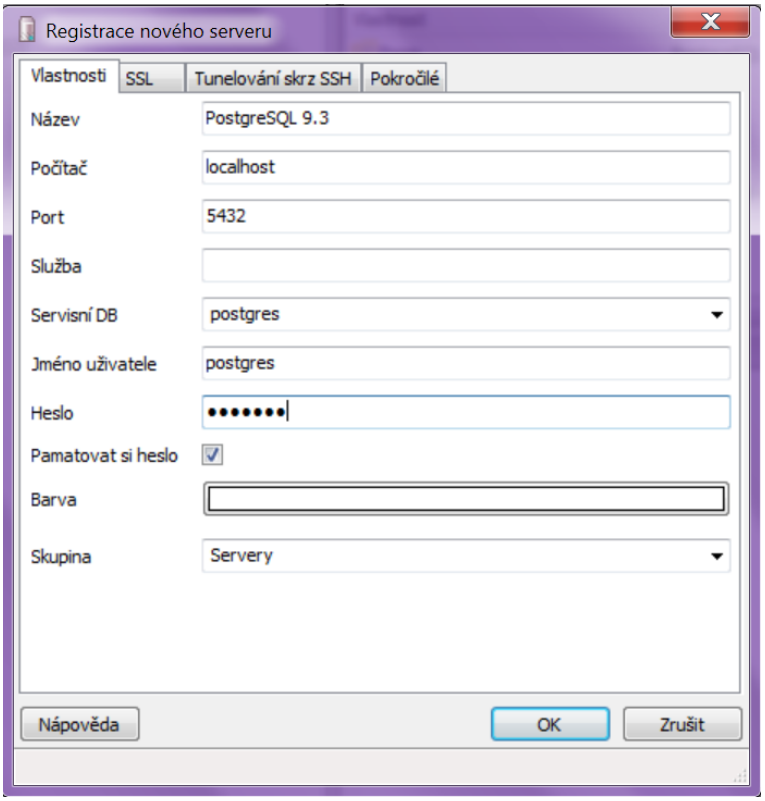
V souboru *setting.xml* lze rovněž nastavit jako jazyk Nástroje češtinu. Pokud jazyk nebude nastaven, bude Nástroj spuštěn v angličtině, která je zvolena jako výchozí jazyk.

A.2 Vytvoření databáze

Je nutné vytvořit databázi pro projekt. Vytvořte databázi „webChangesAnalyser“, která bude patřit roli „webchangesanalyserowner“ s přístupovým heslem „webChangesAnalyserPwd“ a spusťte server localhost na portu 5432. Všechny tyto údaje se dají upravit a změnit v souboru *persistence.xml*. Tuto změnu doporučujeme. V případě, že budete tyto údaje měnit, je potřeba změnit i jméno uživatele „webchanges-analyserowner“ v přiloženém skriptu, aby váš uživatel získal přístupová práva ke správě databázových tabulek.

Poté proveďte v databázi přiložený skript. Skript rovnou vytvoří záznamy o testovacích technikách na sledování změn webových stránek.

Postup pro vytvoření databáze v nástroji pgAdmin: Spusťte nástroj pgAdmin. Pokud nevidíte server PostgreSQL, přidejte do „skupin serverů“ server PostgreSQL. (V hlavním okně zvolte „Soubor“ – „Přidat server“ a vyplňte formulář podle obrázku A.1:

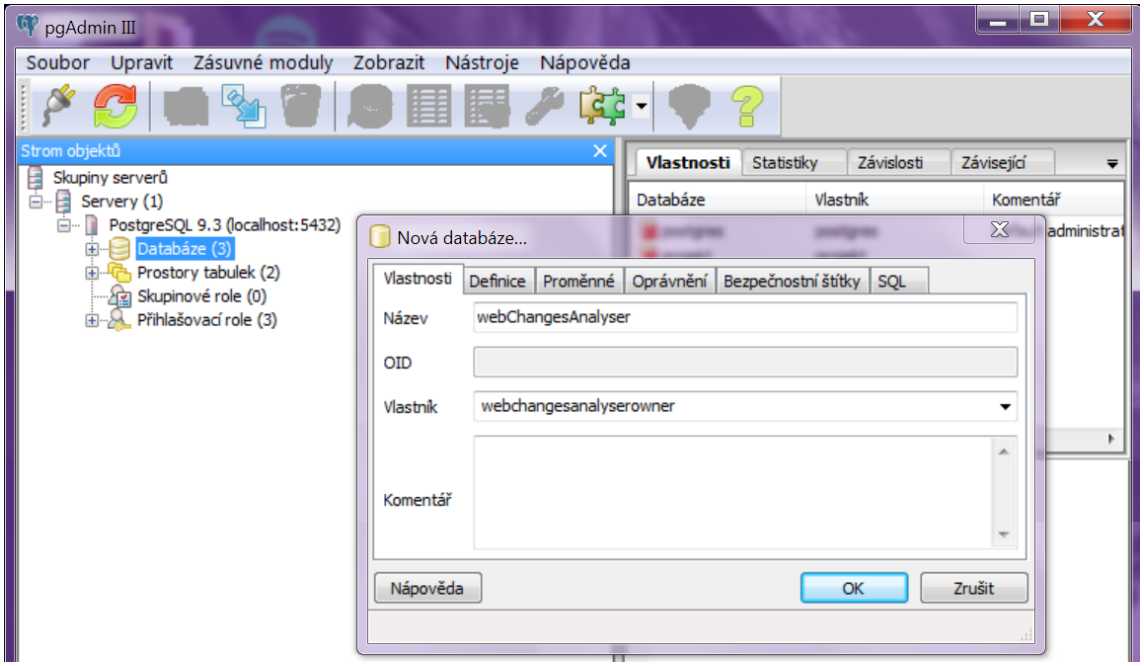


Obrázek A.1. Přidání serveru

Otevřete nabídku skupin serverů a připojte se k serveru PostgreSQL.

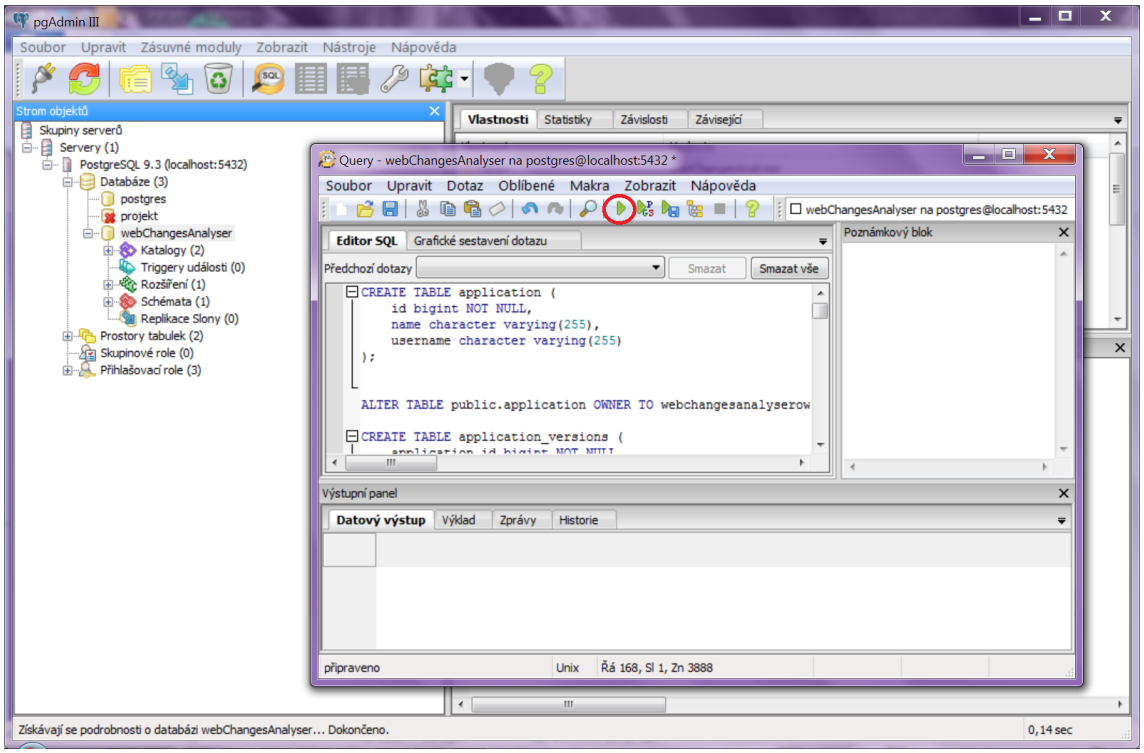
Klikněte pravým tlačítkem na „Přihlašovací role“ a zvolte „Nová přihlašovací role“. V kartě vlastností vyplňte „Název role“ („webchangesanalyserowner“) a v kartě definice zadejte heslo („webChangesAnalyserPwd“). Potvrďte tlačítkem „Ok“.

Poté klikněte pravým tlačítkem na „Databáze“ a zvolte „Nová databáze“. Kartu „Vlastnosti“ vyplňte podle obrázku A.2 a stiskněte „Ok“.



Obrázek A.2. Vytvoření nové databáze

Databáze nyní byla vytvořena. Rozbalte nabídku databází a klikněte dvakrát na jméno databáze „webChangesAnalyser,“ čímž se k databázi připojíte. Poté klikněte v nabídce funkcí na lupu „SQL“ a zobrazí se vám okno pro provádění SQL skriptů. Otevřete přiložený skript pro vytvoření tabulek databáze a tento proveďte stisknutím zelené šipky podle obrázku A.3.



Obrázek A.3. Spuštění skriptu

Nyní je databáze vytvořena včetně struktury a Nástroj můžete spustit.

A.3 Změna údajů pro databázi

Pokud si chcete založit databázi s jinými údaji, bude nutné upravit skript a soubor *persistence.xml*, který se nachází ve složce META-INF.

Je možné použít i jinou databázi než postgres, poté ovšem bude nutné provést více opatření:

1. Upravit skript na jiný druh databáze
2. Přidat do souboru *pom.xml* závislost na tuto databázi místo závislosti na postgres:

```
<dependency>
   <groupId>org.postgresql</groupId>
   <artifactId>postgresql</artifactId>
   <version>9.4-1200-jdbc41</version>
</dependency>
```

3. Upravit soubor *persistence.xml* na hodnoty nového druhu databáze
4. Případně upravit kód v balíčku *dao* tak, aby odpovídal pokynům a dotazům nové databáze.

V komentářích souboru persistence.xml uvádím kam psát změny pro nastavení Postgres databáze. U elementů „property“ s následujícími atributy „name“ můžete měnit obsah atributu „value“ pro požadované nastavení.

Změna ovladače pro Váš druh databáze:

```
name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"
```

Změna portu, na kterém Vaše databáze naslouchá a změna jména Vaší databáze:

```
name="javax.persistence.jdbc.url"
value="jdbc:postgresql://localhost:5432/webChangesAnalyser"
```

Změna jména uživatele Vaší databáze:

```
name="javax.persistence.jdbc.user" value=" webchangesanalyserowner"
```

Změna hesla uživatele Vaší databáze:

```
name="javax.persistence.jdbc.password" value=" webChangesAnalyserPwd"
```

A.4 Kontrola databáze

Skutečnost, že Vaše databáze je propojena s projektem tak, jak má, si můžete ověřit spuštěním JUnit testů z balíčku cz.cvut.fel.service. Pokud oba testy proběhnou v pořádku, můžete již Nástroj bez obav užívat.

Pokud jste vše nastavili správně, bude Vám fungovat i spustitelný jar soubor. Tento soubor lze spustit ze složky, nebo z příkazové řádky příkazem:

```
java -jar cesta_k_souboru.jar
```

Příloha B
Návod k použití

Nástroj pro sledování změn uživatelského rozhraní webové aplikace (dále jen Nástroj) je schopen automaticky uložit zdrojový kód uživatelského rozhraní webové aplikace v několika časových verzích a je schopen tyto verze porovnávat.

Nástroj tímto porovnáváním pomůže testerům webových aplikací analyzovat problémové úseky testované aplikace, jejíž strukturu nemusí předem znát. Nástroj analyzuje dynamické a problémové (např. nevalidní) části aplikace, které by jinak museli testeři zjišťovat manuálně, nebo kombinací více nástrojů. Pro analýzu nabízí Nástroj osm testovacích technik.

Nástroj nabízí osm základních funkcí, které jsou v tomto návodu vysvětleny. Instalace Nástroje a základní nastavení je popsané v Návodu k instalaci A.

Předpokládané použití je takové, že uživatel vytvoří konfigurační soubor pro stahování aplikace, kterou potřebuje analyzovat. Poté uživatel opakovaně provede stahování aplikace na základě konfiguračního souboru, aby získal časové verze, které bude moci porovnávat. Uživatel si sestaví testovací scénář, do kterého zařadí takové testovací techniky, které zjistí změny podle charakteru analyzované aplikace a podle potřeb uživatele. Následně uživatel otestuje stažené verze připraveným scénářem. Stránky aplikace, které se zdají problematické, může uživatel stáhnout opakovaně (bez nutnosti stahování celé aplikace) a otestovat pouze tyto stránky. Stažené aplikace a stránky může uživatel sledovat v hlavním okně.

Pro seznámení s Nástrojem a otestování první aplikace je doporučen tento postup:

1. Vytvoření konfiguračního souboru
2. Crawlování aplikace alespoň ve dvou verzích
3. Vytvoření testovacího scénáře
4. Zvolení aplikace ve stromě aplikací v hlavním okně
5. Otestování aplikace scénářem

B.1 Hlavní okno

Hlavní okno je rozděleno do pěti částí.

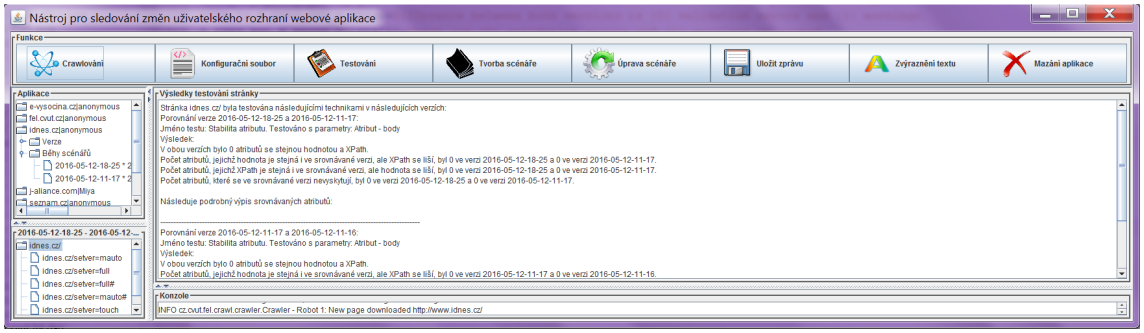
V horní části je nabídka funkcí pro ovládání Nástroje.

V pravé horní části vidíte strom stažených aplikací. Zvolte aplikaci, kterou chcete analyzovat. Po rozbalení větve Vaší aplikace se Vám zobrazí větev s daty stažených verzí a větev s již proběhlými scénáři.

Zvolíte-li nějakou verzi, zobrazí se vám v pravé dolní části strom stránek, které jsou uloženy pro zvolenou aplikaci v dané verzi. Po výběru stránky se Vám zobrazí v levém části zdrojový kód uložené stránky.

Zvolíte-li nějaký proběhlý scénář, zobrazí se vám v pravé dolní části strom stránek, které jsou uloženy pro zvolenou aplikaci v obou verzích, které byly během analyzovány. Po výběru stránky se Vám zobrazí v levé části výsledky všech testování zvolené stránky.

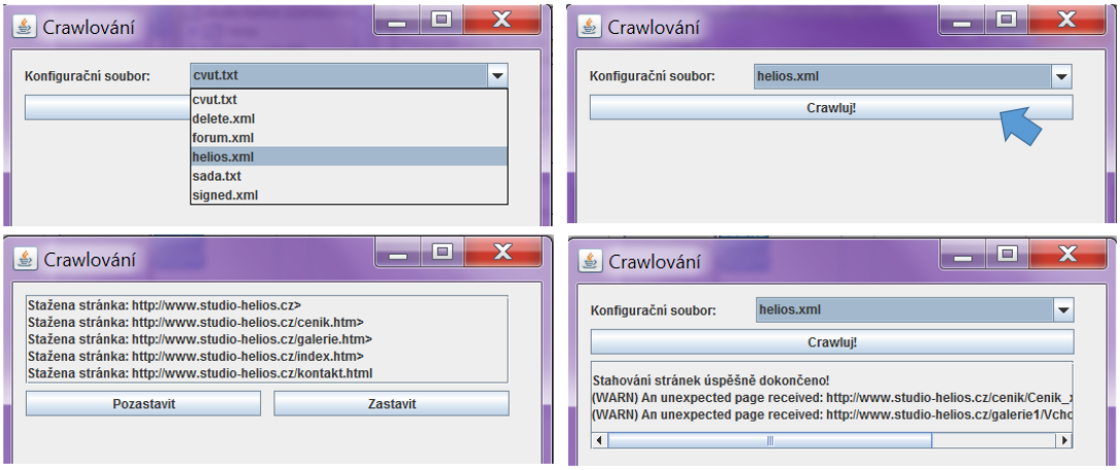
V levé části hlavního okna také Nástroj komunikuje s uživatelem, např. zde zobrazuje výsledky právě dokončeného testování. Navíc je v levé části dole oblast konzole, ve které je uživatel informován o běhu programu. Konzole upozorňuje na průběh crawlování, varuje uživatele a také jej informuje o chybách, které se mohou stát při práci s neočekávanými daty.



Obrázek B.4. Hlavní okno Nástroje

B.2 Crawlování

Tato funkce zahrnuje ukládání stránek k analýze, tedy jak crawlování aplikace, tak stahování sady stránek. Zvolte z nabídky konfigurační soubor a Nástroj spustí stahování či crawlování podle druhu konfiguračního souboru. Můžete sledovat vývoj stahování. Po ukončení Vám Nástroj oznámí výsledek a upozorní Vás na chyby.



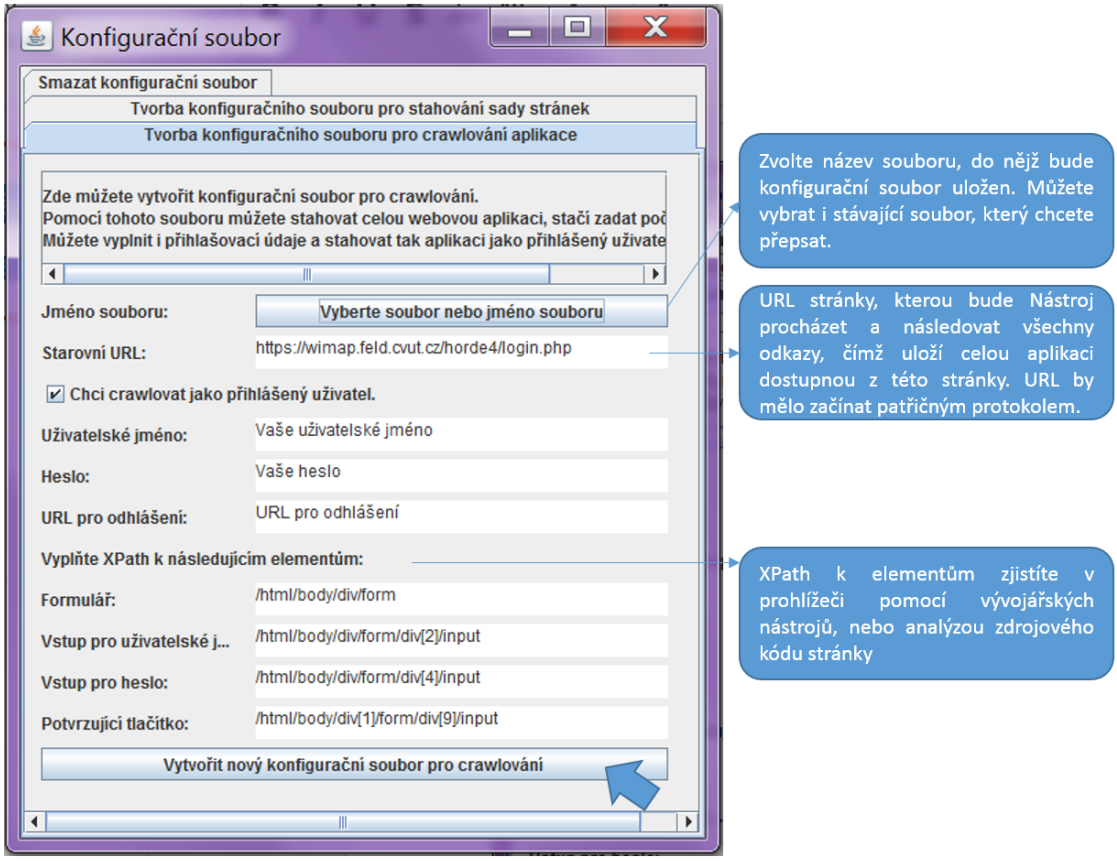
Obrázek B.5. Průběh crawlování aplikace

B.3 Konfigurační soubor

Tato funkce umožňuje vytvoření konfiguračního souboru pro crawlování aplikace či pro stažení sady stránek. Funkce umožňuje také smazat již vytvořené soubory. Tyto možnosti jsou rozděleny do tří karet. Na obrázku B.6 uvádím příklad, jak je možné vytvořit konfigurační soubor pro stažení stránky.

Nástroj kontroluje stránku, která se nachází na adrese startovního URL. Pokud se nemůže Nástroj ke stránce připojit, nebo v jejím obsahu nalezne nějaké podezřelé prvky

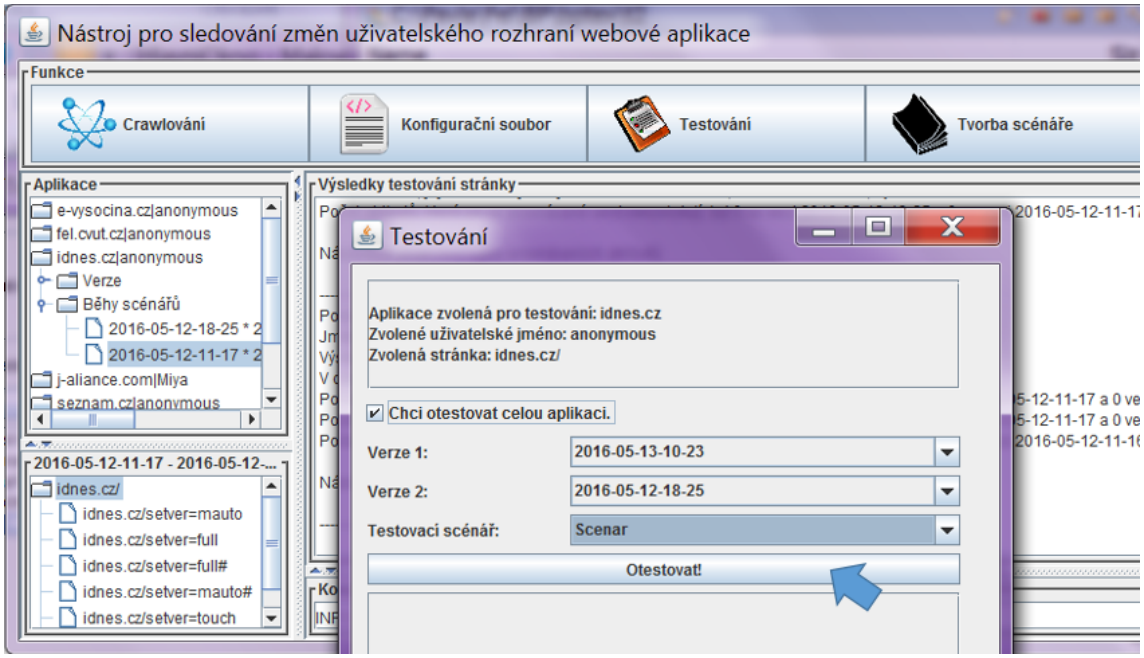
(přesměrování, bannery...), upozorní na to uživatele. Uživatel může konfigurační soubor vytvořit i přes varování, musí ovšem počítat s tím, že uvedenými potíži může být ovlivněné crawlování či testování aplikace.



Obrázek B.6. Tvorba konfiguračního souboru pro crawlování aplikace

B.4 Testování

Tato funkce zahrnuje analýzu uložené stránky či aplikace pomocí definovaného scénáře. Vyberte v hlavním okně aplikaci a případně stránku, vyberte scénář z nabídky Vámi definovaných scénářů a vyberte dvě verze, které chcete porovnávat. Spusťte testování. Nástroj Vás bude informovat o tom, kolik stránek již analyzoval. Na závěr Vám zobrazí zprávu, ve které se dozvíte informace potřebné ke srovnání stránky.



Obrázek B.7. Okno pro spuštění testování aplikace

Uvádím smyšlený příklad výsledné zprávy pro aplikaci a pro jednu stránku pro testovací techniku na počet validačních chyb a varování:

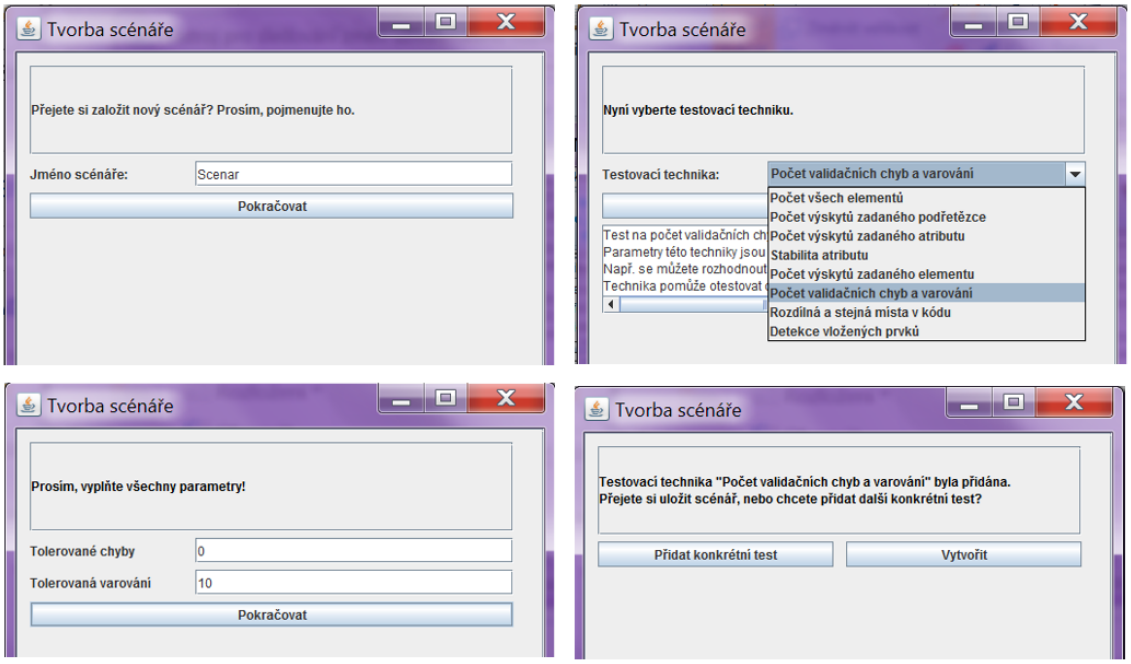
```
Aplikace ve verzi 2016-04-18-23-29 obsahuje 507 validačních chyb a 172
varování.  Aplikace obsahuje 0 validních a 8 nevalidních stránek. 0%
aplikace je validní.
Aplikace ve verzi 2016-05-05-15-51 obsahuje 302 validačních chyb a 50
varování.  Aplikace obsahuje 4 validních a 5 nevalidních stránek. 44%
aplikace je validní.
Tolerováno je 0 validačních chyb a 10 varování.
Ve verzi 2016-04-18-23-29 v porovnání s verzí 2016-05-05-15-51 chybí 2
stránek.
Ve verzi 2016-05-05-15-51 v porovnání s verzí 2016-04-18-23-29 chybí 0
stránek.

Název-stránky
Stránka ve verzi 2016-04-18-23-29 obsahuje 52 validačních chyb a 20 varo-
vání. Stránka není validní.
Stránka ve verzi 2016-05-05-15-51 obsahuje 0 validačních chyb a 4 varo-
vání. Stránka je validní.
Rozdíl mezi oběma verzemi je 52 validačních chyb a 16 varování.
Tolerováno je 0 validačních chyb a 10 varování.
```

B.5 Tvorba scénáře

Tato funkce Vám umožní vytvořit testovací scénář a přidat do něj libovolné množství definovaných testovacích technik s Vašimi parametry. Scénářem můžete poté testovat Vaše stažené aplikace.

Do testovacího scénáře je možné přidat konkrétní testy. Testy vychází z připravených technik, které vyhodnotí různé aspekty stránky a porovnají tento aspekt v různých verzích stránky. Techniky, které vyžadují vyplnění parametrů, lze do scénáře přidat vícekrát s různými parametry.



Obrázek B.8. Průběh tvorby testovacího scénáře

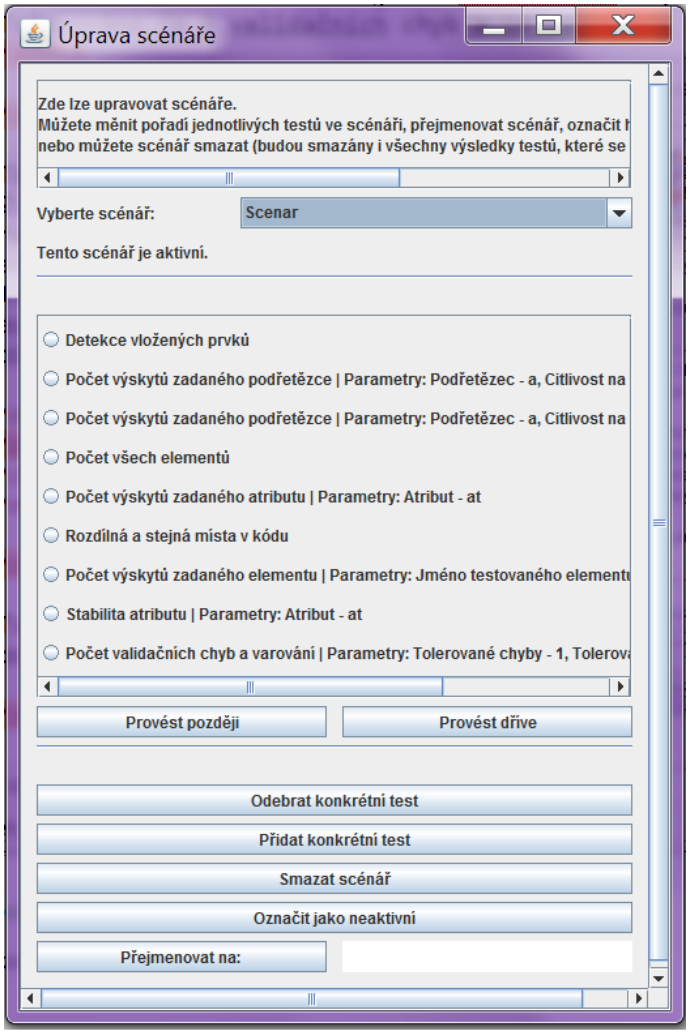
Nástroj nabízí tyto testovací techniky:

1. **Test na počet všech elementů** spočítá, kolik elementů (HTML tagů) je v kódu stránky. Technika spočítá úplně všechny elementy (počínaje kořenovým elementem dokumentu až po ty nejmenší) a navíc porovná, zda se element se stejným obsahem vyskytuje i ve srovnávané verzi. Pomůže tak uživateli získat nadhled, do jaké míry se v porovnání změnila velikost stránky a jak velká část elementů byla změněna.
2. **Test na počet výskytů zadaného podřetězce** vyhledá všechny výskyty zadaného podřetězce v kódu stránky. Jako podřetězec může být zadán text (např. klíčové slovo z tématu testované aplikace) i regulární výraz. Parametry této techniky jsou dva – podřetězec a citlivost na velikost písmen („ano“ nebo „ne“). Technika se zaměřuje především na zkoumání konkrétního problému s nějakým textem.
3. **Test na počet výskytů zadaného elementu** vyhledá, kolikrát se v kódu vyskytuje element s daným názvem (tagem). Parametrem této techniky je jméno elementu (např. img, a, p...) Uživatel touto technikou může sledovat změnu v počtu výskytů podezřelého elementu.
4. **Test na počet výskytů zadaného atributu** vyhledá, kolikrát se v kódu vyskytuje atribut s daným názvem. Parametrem tohoto testu je jméno atributu (např. id, class, name). Uživatel touto technikou může sledovat změnu v počtu výskytů podezřelého atributu.

5. **Test na stabilitu atributu** vyhledá všechny atributy s daným názvem, porovná tyto atributy v obou verzích a rozřadí je do čtyř skupin podle hodnoty a XPath atributu. Parametrem tohoto testu je jméno atributu (např. id, class, name). Uživatel touto technikou může sledovat, jak se podezřelý atribut měnil (např. jestli je hodnota atributu „id“ vždy stejná, nebo je generovaná dynamicky). Protože technika pracuje s XPath atributů, může být citlivá na příliš nevalidní a problematické stránky.
6. **Test na počet validačních chyb a varování** zjistí počet validačních chyb a počet varování pro stránku podle kritérií W3C validátoru. Parametry této techniky jsou počet tolerovaných chyb a počet tolerovaných varování. Např. se můžete rozhodnout pro 0 validačních chyb, ale povolit 10 varování, aby byla stránka stále považována za validní. Technika pomůže otestovat chybovost zdrojového kódu stránek aplikace a uživatel získá přehled o tom, jak problematická je testovaná aplikace.
7. **Test na rozdílná a stejná místa v kódu** porovná zdrojový kód stránky ve dvou časových verzí znak po znaku a vyhledá podřetězce, ve kterých se zdroje stránek liší a ve kterých se shodují. **Varování:** tato technika může být časově náročná pro velké stránky, doporučuji tedy techniku používat až pro detailní analýzu jednotlivých stránek, ne pro analýzu celé aplikace. Technika zvýrazní místa změn a uživatel tak snáze uvidí, na kterých místech došlo ke změnám.
8. **Test na výskyt vložených prvků** má seznam regulárních výrazů v souboru embeddedObjects.xml, který může uživatel upravit sám podle svých potřeb. Nástroj na základě seznamu spočítá výskyt prvků, které komplikují automatické testování, jakými jsou např. vložené aplikace, multimediální obsah a jiné. Uživatel si pomocí této techniky sám definuje problematické prvky, na které se chce zaměřit, nebo může sledovat prvky, které ovlivňují podobu aplikace (např. javascripty, vnořené Flashové aplikace, javovské applety a další).

B.6 Úprava scénáře

Tato funkce Vám nabízí úpravu, zobrazení a mazání již vytvořených scénářů. Zvolíte-li scénář z nabídky, zobrazí se Vám, zda je aktivní (tzn. že se zobrazuje v nabídce scénářů pro testování), zobrazí se Vám seznam konkrétních testů, které scénář obsahuje, a to v pořadí, ve kterém testy probíhají. Můžete měnit pořadí těchto testů, můžete přidat nový konkrétní test, nebo můžete konkrétní testy ze scénáře odebrat, až na poslední. Můžete smazat také celý scénář. Scénář navíc můžete označit jako aktivní či neaktivní a můžete ho přejmenovat.



Obrázek B.9. Okno se zobrazením, úpravou a mazáním testovacích scénářů

B.7 Uložení zprávy

Tato funkce načte zprávu z hlavního okna a uloží ji do textového souboru, který pojmenujete, nebo přepíše soubor, který již existuje.

B.8 Zvýraznění textu

Tato funkce otevře vždy nové okno se zprávou z hlavního okna. Díky tomu můžete např. otevřít vedle sebe výsledky testování stránky a zdrojové kódy v porovnávaných

verzích. Funkce formátuje zprávu z hlavní okna podle toho, co bylo zobrazené. Pokud byl zobrazený zdrojový kód, zvýrazní syntax HTML kódu. Pokud byly zobrazeny výsledky testu, podtrhá jména testovacích technik a zvýrazní porovnávané verze. Funkce také pomůže při zobrazení výsledků testovací techniky Test na rozdílná a stejná místa v kódu, tím, že rozdílná místa obarví dvěma různými barvami.

B.9 Mazání aplikace

Pokud již nebudete chtít nadále s nějakou aplikací pracovat, nebo budete chtít smazat staré verze, funkce Vám nabídne seznam aplikací a Vy můžete smazat libovolnou aplikaci, nebo můžete smazat všechny verze, které jsou starší než Vámi vybraná verze. Nástroj smaže aplikace či staré verze včetně výsledků analýzy.

Příloha C
Obsah přiloženého CD

Přiložené CD obsahuje tyto soubory:

- Tuto bakalářskou práci včetně příloh A a B ve formátu PDF
- Zdrojový kód a obrázky k této bakalářské práci
- Implementovaný Nástroj jako spustitelný soubor .jar
- Zdrojový kód Nástroje
- Sestavený Netbeans projekt