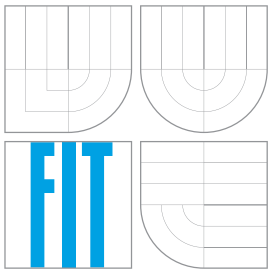
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ÚPRAVA A VYLEPŠENÍ NÁSTROJE SELFTEST

SELFTEST TOOL IMPROVEMENTS AND FIXES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

OTTO ŠABART

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ALEŠ SMRČKA, Ph.D.

BRNO 2014

Abstrakt

SelfTest je nástroj určený k testování znalostí lidí na základě předem vytvořených testovacích sad. Program do současné doby nedisponoval vhodnými prostředky pro vyhodnocování znalostí účastníků testu a analýzu testovacích sad. Tato práce se zabývá především jejich návrhem a popisem implementace. Dále jsou v práci rozebrány skripty pro vyhodnocování parametrů testovací sady a navrženy různé metriky pro hodnocení kvality účastníků a sady otázek. Práce také zmiňuje některé další části programu, které bylo nutné nebo vhodné implementovat navíc. Byla přidána podpora pro databázi SQLite3, vytvořen databázový adaptér, vylepšena tvorba záznamů událostí a upraveno rozhraní některých tříd.

Abstract

SelfTest is a tool designed to test user knowledge on the basis of preformed testing sets. This software hasn't had any user quality evaluation tools nor testing set analysis tools. Therefore this thesis covers design and description of its implementation. It also examines scripts for testing set parameters evaluation and designes various metrics for user quality and testing sets evaluation. Adding support for SQLite3 database, creating database adapter, improving logging and adjusting some class interfaces are some other parts of the program also mentioned in this thesis, which were necessary (or appropriate) to implement additionaly.

Klíčová slova

SelfTest, testovací nástroj, hodnocení kvality účastníků, parametry testovací sady otázek, metriky hodnocení kvality sady otázek, zobrazování výsledků testu

Keywords

SelfTest, testing tool, user quality evaluation, testing sets parameters, metrics of testing sets quality evaluation, displaying test results

Citace

Otto Šabart: Úprava a vylepšení nástroje SelfTest, bakalářská práce, Brno, FIT VUT v Brně, 2014

Úprava a vylepšení nástroje SelfTest

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Aleše Smrčky, Ph.D. Další informace mi byly poskytnuty panem Janem Hutařem ze společnosti Red Hat. Uvedl jsem všechny publikace a prameny, ze kterých bylo při psaní čerpáno.

.....
Otto Šabart
20. května 2014

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Aleši Smrčkovi, Ph.D a Janu Hutařovi z firmy Red Hat za odborné vedení a připomínky, které mi při psaní práce poskytli.

© Otto Šabart, 2014.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

SelfTest je nástroj určený pro testování, popř. získávání znalostí lidí na základě předem vytvořených dotazníků (testovacích sad). Cílem této bakalářské práce je ukázat jak byl projekt SelfTest rozšířen za účelem analýzy a vyhodnocování znalostí účastníků testu a vyhodnocování různých parametrů otázek. Dále je v práci popsán návrh a postup implementace různých metrik pro hodnocení kvality účastníků a sad otázek. Je zde také popsaná implementace různých skriptů, které byly napsány za účelem pohodlnější práce při znovuvyhodnocování odpovědí účastníků při pozdějších úpravách testovacích sad. Jako poslední jsou zde popsána různá další rozšíření, která bylo nutné vytvořit především kvůli požadavkům společnosti Red Hat[1], která tento software primárně vyvíjí. Jedná se o přidání podpory databáze SQLite, změnu systému pro záznam událostí apod. (více v příloze B). Při vývoji také docházelo k rozsáhlému refaktorování kódu. Byl odstraněn duplicitní kód a zjednodušeny některé nepřiměřeně dlouhé metody. Dále byly některé procedurální části kódu přepsány do kódu objektového. Jednotkové testy, které již kód programu obsahoval, byly upraveny a doplněny o nové.

V současné době je tento program používán především u příležitostí pořádaných firmou Red Hat jako například *Fedora Release Party*, *Software Freedom Day*, či *OpenHouse*. Na těchto akcích se využívá především při konání znalostních soutěží *BashMaster* a *CMaster* jako testovací nástroj, kde se testují znalosti uživatelů v oblasti informačních technologií. Tento nástroj lze ale využít i pro testování znalostí z jiných oblastí. Dále je program touto společností využíván pro běh dotazníku, který slouží pro pohodlnější rozhodování při přijímání nových zaměstnanců.

Projekt je vyvíjen komunitně od roku 2009, kdy vyšel ve své první verzi *v0.5*. Od té doby je vyvíjen pod licencí GPLv2[2] v rámci společnosti Red Hat. Jeho repozitář je veřejně přístupný[3].

[1] Viz http://www.redhat.com

[2] *GPLv2 (General Public Licence, version 2)* - licence pro svobodný software.

[3] Viz https://fedorahosted.org/selftest

Kapitola 2

Nástroj SelfTest

Kapitola se zabývá popisem testovacího nástroje SelfTest. Popisuje technologie, které jsou pro jeho běh vyžadovány, ukazuje jak vypadá zpracování klientského požadavku. Dále představuje jeho adresářovou a modulární strukturu, vysvětluje jednotlivé prvky testovací sady jako oblasti a otázky a rozebírá jejich konfigurační soubory. Na konci kapitoly je program SelfTest porovnán s podobnými alternativními nástroji pro testování a vytváření dotazníků, které jsou v současné době na trhu.

2.1 Popis nástroje

Jedná se o nástroj fungující jako webový server. S ním pak účastník testu komunikuje pomocí webového prohlížeče. Nástroj na základě předem připravené testovací sady nabízí uživatelům otázky. Na obrázku č. 2.1 je zachycen úvodní vzhled programu. Takto program vidí účastník testu předtím, než je test spuštěn.

2.2 Použité technologie

Celý nástroj SelfTest i výsledné skripty jsou napsány v jazyce Python verze 2. Komunikaci na úrovni protokolu HTTP zajišťuje modul WSGI (viz podkapitola 2.2.1). Pro zobrazování odpovědí serveru uživateli je použito HTML ve spojení s kaskádovými styly (CSS) a jazykem JavaScript - konkrétně knihovnou jQuery[1]. Ta je v současné době využívána pouze pro zobrazování ukazatele zbývajícího času, který má uživatel na svou odpověď. V budoucnu je ale pro ni plánováno další využití.

Databázové rozhraní dříve podporovalo pouze dotazy nad tabulkovými daty typu CSV[2]. V rámci této bakalářské práce bylo upraveno databázové rozhraní tak, aby bylo nezávislé na použité databázi a zároveň bylo rozšířeno o podporu SQLite3. Více lze nalézt v příloze B.1.

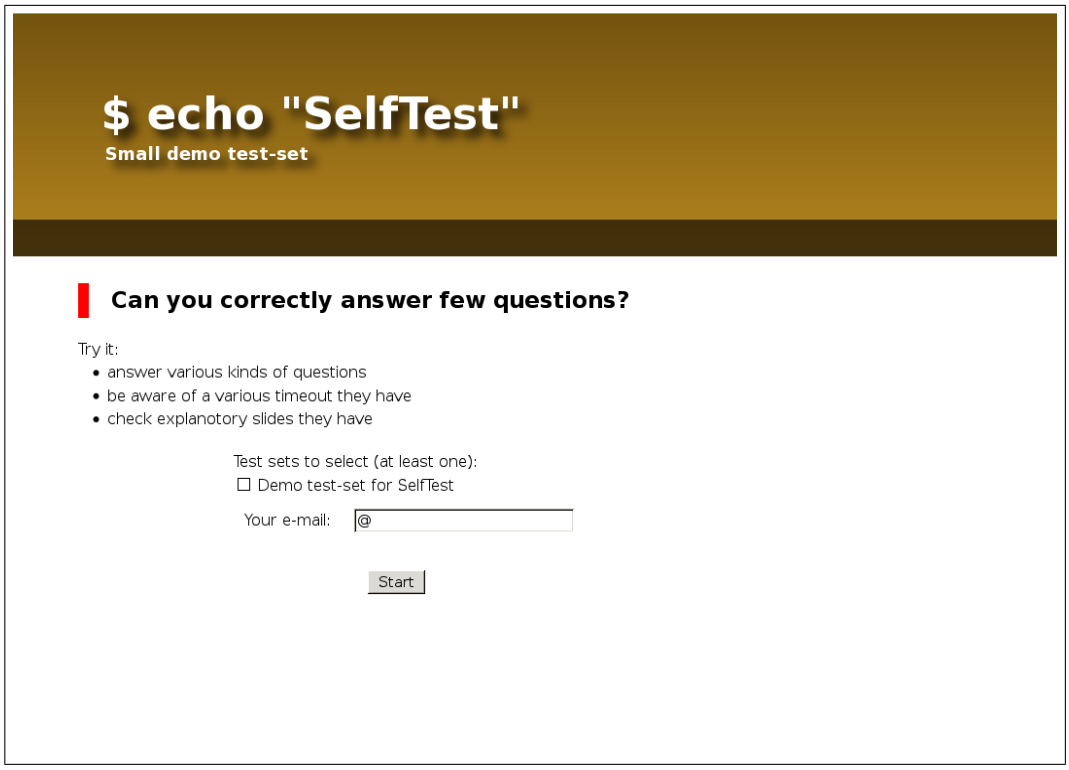
2.2.1 Python

Jedná se o vysokoúrovňový multiplatformní dynamický interpretovaný jazyk, který je objektově orientovaný [9]. Je distribuován pod otevřenou licencí Python Software Foundation Licence[3].

[1] Viz http://jquery.com

[2] *CSV (Comma-separated values)* - jednoduchý souborový formát určený pro výměnu tabulkových dat.

[3] Viz http://docs.python.org/2/license.html



Obrázek 2.1: Ukázka úvodní strany dotazníku.

Modul WSGI (Web Server Gateway Interface)

Komunikace mezi klientem a serverem v aplikaci SelfTest probíhá prostřednictvím modulu `wsgiref`, který je součástí standardní knihovny jazyka Python. Tento modul poskytuje rozhraní mezi softwarem webového serveru a web aplikacemi napsanými v jazyce Python. Díky tomu, že toto rozhraní je standardizované [11], je možné vytvářet aplikace, které poběží na různých webových serverech. Není tak potřeba znát konkrétní detaily implementace WSGI. K napsání webové aplikace stačí pouze využít některý z existujících frameworků [10].

Knihovna matplotlib

Jedná se o knihovnu pro vykreslování velmi kvalitních 2D grafů. Podporuje export do mnoha různých formátů. Tuto knihovnu lze používat nejen ve skriptech jazyka Python, ale i v příkazovém procesoru *IPython* nebo na webových aplikačních serverech. Je distribuována pod licencí BSD [5].

2.2.2 SQLite3

SQLite3 je multiplatformní relační databázový systém, který je šířen pod svobodnou licencí *Public domain* [12]. Tento systém není založen na architektuře klient-server (není zde žádný databázový server jako u většiny ostatních relačních databází). Čtení i zápis z/do databáze probíhá nad klasickými soubory na disku. Jedna databáze je představována jedním souborem, ve kterém se nacházejí jednotlivé součásti relační databáze jako tabulky, indexy, triggery a pohledy.

SQLite3 je především knihovna, kterou lze pomocí jednoduchého rozhraní přímo používat. Tento databázový systém je nenáročný na velikost. Velikost celé knihovny nepřesahuje 500 KiB [12]. Jelikož pro běh není potřeba server, využívá se tato databáze v mnoha počítačových programech. Příkladem těchto programů mohou být *Skype*, *Dropbox Client*, *Firefox* nebo *Adobe Acrobat Reader* [13].

Databázi SQLite je možné používat prostřednictvím mnoha programovacích jazyků. Najdeme ji například v C, C++, C#, PHP, Python, Perl, Ruby nebo Javě.

2.3 Popis funkcionality programu SelfTest

Jedná se o aplikaci běžící jako webová služba, která pro svůj běh využívá rozhraní WSGI.

Průběh testování

Při prvním příchodu je uživateli zobrazena úvodní stránka (viz obr. č. 2.1), kde je možné si vybrat jaké testovací sady mu budou v průběhu testování nabídnuty. Dále je po uživateli vyžadována jeho e-mailová adresa, která v SelfTestu slouží jako jednoznačný identifikátor uživatele. Po potvrzení je předem vygenerován seznam otázek, které budou v průběhu testu uživateli nabídnuty. Následně je načtena konfigurace první otázky z testovací sady, která je na jejím základě uživateli zobrazena. Takto probíhá celé testování dokud uživateli nejsou nabídnuty všechny otázky. Jako poslední je uživateli zobrazena konečná úspěšnost v testu a výsledky jednotlivých otázek.

2.3.1 Interní zpracování klientského požadavku

Na obrázku 2.2 můžeme vidět grafické znázornění zpracování požadavku klienta zaslaného na server.

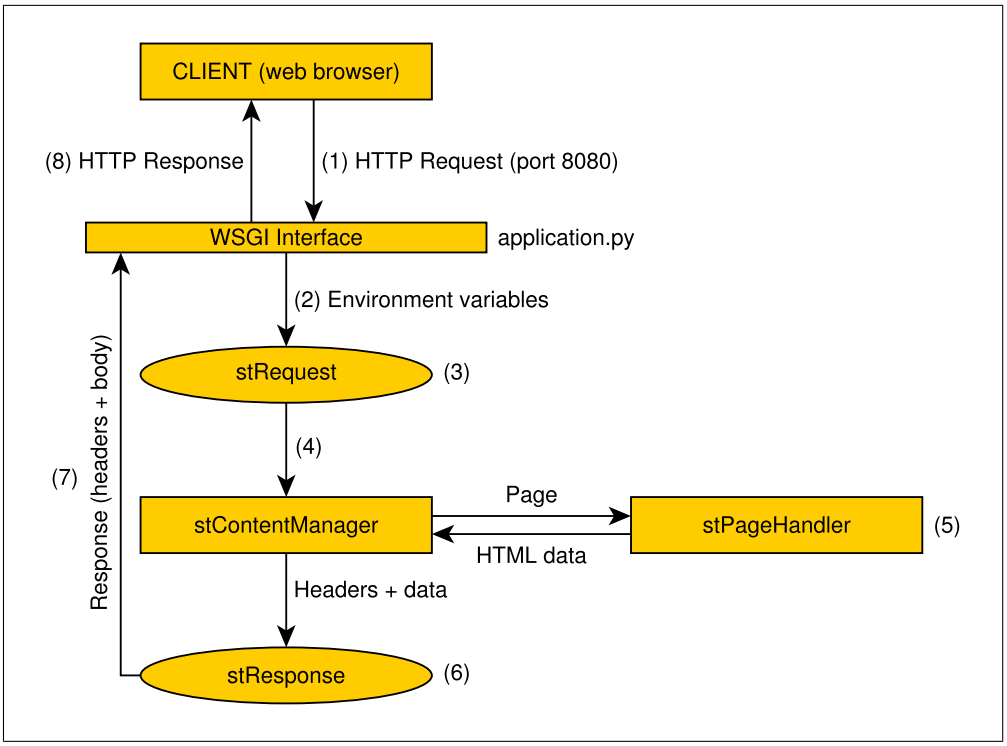
Při spuštění programu SelfTest pomocí skriptu `application.py` se jako první spustí WSGI server, který otevře port (v případě programu SelfTest port 8080), na kterém naslouchá příchozí HTTP[4] požadavek. Ukázku vytvoření WSGI serveru, který zpracovává klientský požadavek pomocí aplikace `demo_app`, lze vidět na ukázce kódu č. 2.1.

```
1  # Import potřebných balíčků
2  from wsgiref.simple_server import make_server, demo_app
3
4  # Vytvoření WSGI serveru, který bude naslouchat na portu 8080
5  # a požadavek zpracovávat pomocí aplikace demo_app
6  httpd = make_server('', 8080, demo_app)
7  httpd.serve_forever()
```

Kód 2.1: Ukázka vytvoření WSGI serveru v jazyce Python2.

Jakmile požadavek z klienta přijde (1), rozhraní ho zpracuje a řízení předá samotné aplikaci, které zároveň obdrží i slovník proměnných (2). Přesná podoba tohoto slovníku je specifikována [11]. Z těchto proměnných lze pak v aplikaci vyčíst mnoho detailních informací o HTTP hlavičkách požadavku, na základě kterých se aplikace zachová různě. Vyčíst můžeme různé části URL (proměnné `SCRIPT_NAME`, `PATH_INFO`, `QUERY_STRING`), typ metody, kterou byla URL zavolána (proměnná `REQUEST_METHOD`), nebo verzi komunikačního protokolu (`SERVER_PROTOCOL`).

[4] *HTTP (Hypertext Transfer Protocol)* - internetový protokol pro výměnu hypertextových dokumentů formátu HTML.



Obrázek 2.2: Zpracování klientského požadavku.

Pro lepší interní práci s těmito daty obsahuje program třídu `stRequest` (3). Ta data zaobaluje a pro práci s nimi poskytuje intuitivnější rozhraní. Tato třída je předána objektu `stContentManager` (4), který má za úkol generovat odpověď serveru. To činí na základě již zmiňovaných proměnných. V případě, že klient požaduje stránku, kterou aplikace nenabízí, navrátí odpověď typu „404 - strana nenalezena“. Jinak za pomoci třídy `stPageHandler` získá HTML data webové stránky (5), která budou odeslána zpět na klienta. K těmto datům přidá patřičné HTTP hlavičky a vše zabalí do objektu `stResponse` (6). Ten je následně předán rozhraní WSGI (7), které už se samo postará o odeslání všech dat pomocí protokolu HTTP zpět na klienta (8).

2.4 Modulární struktura

Celý nástroj se skládá z modulů, které obsahují funkce a třídy představující objekty programu. Objekty jsou mezi sebou vzájemně propojené.

Nejvýznamnějšími moduly programu jsou:

- `questions` – Obsahuje třídy představující otázky (stQuestion), oblasti (stArea) nebo celé testovací sady (stTestSet).
- `users` – Obsahuje třídu `stUser`, představující účastníka testu.
- `database` – Tento modul obsahuje implementaci databázového adapteru a třídu `DbManager` pro práci s databází.
- `pages` – Obsahuje především třídu `stPageHandler`, která zajišťuje získávání dat HTML při odpovědi serveru.

- `utils` – Zde se nacházejí různé pomocné funkce jako funkce pro odstraňování HTML elementů, ukládání statických výsledků uživatelů apod.
- `communication` – Obsahuje třídy, které představují klientský požadavek (`stRequest`) a odpověď serveru (`stResponse`).
- `results` – Jejím obsahem jsou třídy, které umožňují přívětivé získávání uživatelských výsledků z databáze.

2.5 Adresářová struktura

Kořenový adresář projektu obsahuje několik důležitých adresářů a souborů. Následující část jednotlivé adresáře a soubory popisuje.

Adresáře

- `src` – Obsahuje zdrojové soubory programu.
- `tests` – V tomto adresáři se nacházejí podadresáře obsahující soubory testovacích sad.
- `database` – Adresář obsahující databázové soubory.
- `res` – Zde se nacházejí soubory klientské části jako obrázky, CSS a JavaScript.
- `results` – Obsahuje statické soubory výsledků uživatelů.
- `tools` – Jsou zde pomocné skripty pro analýzu výsledků uživatelů a testovacích sad.

Soubory

- `application.py` – Tímto souborem je spouštěn WSGI server nástroje SelfTest.
- `make-test.sh` – Skript testující otázky typu *„exec“*.
- `makeopenshift.sh` – Skript pro vytvoření takové struktury programu, kterou je možné přenést do cloudu *OpenShift*[5].
- `selftest.service` – Konfigurační soubor služby pro *SystemD*[6].

Konfigurační soubor nástroje SelfTest

Hlavním konfiguračním souborem je `config.py`, který se nachází v adresáři `src`. Umožňuje nastavovat různé vlastnosti nástroje a měnit tak jeho chování. Je zde možné nastavit úroveň či formát výpisu programových hlášení nebo zvolit mezi databázemi CSV a SQLite. Dále umožňuje nastavení odesílání výsledků na e-mail prostřednictvím SMTP[7] nebo měnit heslo do administrátorské sekce.

[5] *OpenShift* - Cloud computingová platforma nabízená jako produkt společnosti Red Hat.
[6] *SystemD* - linuxový init systém.
[7] *SMTP (Simple Mail Transfer Protocol)* - internetový protokol pro přenos zpráv elektronické pošty.

2.6 Struktura testovací sady

Testovací sadu představuje adresář, ve kterém se musí nacházet soubor `testset.config` (viz kód na obrázku 2.2). V něm je pak základní konfigurace sady.

```
1 [testset]
2 message_1_when_ge = 50
3 message_1_text = @message_1.html
```

Kód 2.2: Ukázka konfigurace testovací sady.

Volby `message_1_when_ge` a `message_1_text` říkají, jaká zpráva se zobrazí uživateli po dokončení testu, když bude mít konečnou úspěšnost vyšší jak 50%. V tomto případě je zpráva načtena z HTML souboru `message_1.html`.

Dále se v tomto adresáři mohou nacházet soubory:

- `badges` – Specifikace „hodností“, kterých uživatel může dosáhnout podle toho jak bude úspěšný v testu.
- `description` – Obsahuje název testovací sady, který se zobrazuje uživatelům na úvodní straně.
- `welcome` – Je v něm text, který se zobrazuje uživateli po výběru testovací sady před samotným spuštěním testu.
- `res` – Adresář, ve kterém se nacházejí další dodatečné zdroje související přímo s danou testovací sadou jako obrázky, CSS apod.

Dále každá testovací sada musí kromě konfiguračního souboru obsahovat podsložky. Ty představují tzv. *oblasti*. V těchto oblastech se nacházejí jednotlivé otázky.

2.6.1 Oblasti otázek

SelfTest podporuje rozdělení otázek do tzv. *oblastí*. Z těchto oblastí jsou v průběhu testu uživateli generovány otázky. Neslouží k vyhodnocování testu. Slouží výhradně k logickému členění průběhu testování. Oblast si lze představit jako skupinu otázek nějaké jedné vlastnosti. Příkladem takové vlastnosti může být obtížnost nebo typ otázky. V průběhu testování jsou otázky z těchto skupin vybírány (viz dále). Otázka může vždy náležet pouze do **jedné** oblasti - ne více.

Každou oblast představuje podsložka v testovací sadě, ve které se nachází soubor `questions_to_test.txt`. Obsahem tohoto souboru je pouze přirozené číslo, které udává kolik otázek se z dané oblasti účastníkovi testu **náhodně** vybere.

Výběr otázek z oblasti

Mějme oblast a. V této oblasti budeme mít množinu otázek Q_a. Dále zde bude konstanta K_a, která bude představovat hodnotu *„questions to test“* pro oblast a. Pak:

- $K_a < |Q_a|$ – Z množiny Q_a je uživatelovi náhodně nabídnuto pouze K_a otázek.
- $K_a == 1 \wedge |Q_a| > 1$ – Z množiny Q_a je uživateli náhodně vybrána pouze 1 otázka. Ostatní otázky z množiny nebudou uživateli vůbec nabídnuty.

- $K_a \geq |Q_a|$ – Z množiny Q_a jsou uživatelovi náhodně vybrány vždy všechny otázky. Každá však pouze **jednou**.

Reálný příklad použití oblastí:

Mějme tři oblasti otázek. V první se nacházejí otázky typu *„abc“*, ve druhé otázky typu *„exec“* a ve třetí otevřené otázky typu *„noscore“*. Dále mějme test, kde každému uživateli budeme chtít postupně nabídnout dvě náhodně vybrané otázky z oblasti s otázkami typu *„abc“*, tři z oblasti s otázkami *„exec“* a jednu z oblasti otázek typu *„noscore“*. Struktura oblastí a jejich otázek by v tomto případě vypadala následovně:

```
otazky_abc: [questions to test = 2]
    1, 2, 3, 4, ... N
otazky_exec: [questions to test = 3]
    1, 2, 3, 5, ... N
otazky_noscore: [questions to test = 1]
    1, 2, 3, 6, ... N
```

Vedle jména oblasti je v hranaté závorce vždy uvedena hodnota uložená v souboru `questions_to_test.txt`. Čísla pod jménem oblasti pak představují jednotlivé otázky oblasti.

2.6.2 Interní struktura otázky

V terminologii programu SelfTest je otázka pouze další podadresář adresáře oblasti. Tento adresář musí obsahovat konfigurační soubor otázky `config` (viz ukázka konfigurace otázky na obrázku č. 2.3).

```
[question]
type=<typ otázky>
question=<text otázky v HTML>
timeout=<počet sekund - čas odpovědi>
answers=<pole odpovědí>
correct=<správná odpověď>
categories=<okruhy otázky oddělené čárkou>
```

Kód 2.3: Ukázka konfigurace otázky.

Konkrétní příklady konfigurací otázek lze najít v příloze C.

Otázka s více odpověďmi „abc“

Jedná se o uzavřený typ otázky, ve které jsou předem dané odpovědi. Uživatel má za úkol vybrat správnou odpověď. SelfTest prozatím nepodporuje otázky, u kterých je více správných odpovědí.

Otázka typu „grep“

Tyto otázky jsou uzavřené. Uživatel do textového pole vyplňuje slovo popř. slova. Tento vstup je následně kontrolován pomocí regulárního výrazu, který je specifikovaný v konfiguraci otázky. Regulárním výrazem je tedy definována množina správných odpovědí.

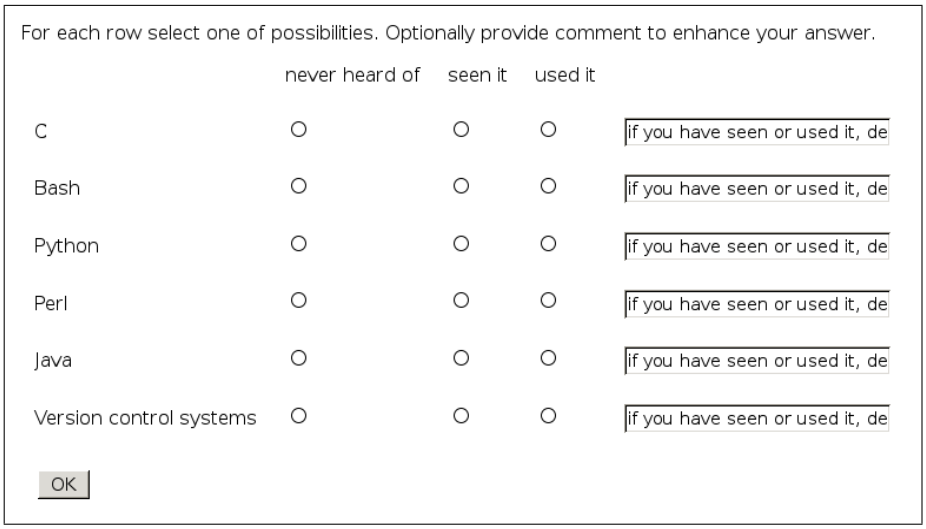
Skriptovatelná otázka „exec“

U těchto otázek je správnost odpovědi ověřována pomocí předem definovaného programu. Tomuto programu, popř. skriptu je uživatelova odpověď předána jako argument. Program při ukončení vrací nulový stav v případě korektní odpovědi, nenulový stav vrací v případě odpovědi chybné.

Maticová otázka „matrix radio comment“

Jedná se o uzavřenou otázku, která je podobná otázce s více odpověďmi. Je vykreslována jako tabulka. Na řádcích se nacházejí otázky a v jednotlivých sloupcích odpovědi. Maticové otázky se používají především v případech, kdy máme více různých otázek se stejnými odpověďmi. Nástroj SelfTest navíc umožňuje každou z otázek doplnit o tzv. komentář. Ukázku otázky lze nalézt na obrázku 2.3.

Tento typ otázky není nástrojem SelfTest nijak vyhodnocován, pouze se zaznamenávají vyplněná data pro budoucí analýzu. Jedná se tak o variantu nehodnocené otázky, která se uplatní hlavně při vytváření dotazníků.



Obrázek 2.3: Ukázka maticové otázky.

Nehodnocená otevřená otázka „noscore“

Tento typ otázky umožňuje uživateli vyjádřit se **svými vlastními slovy**. Nedostává na výběr z předpřipravených variant odpovědí. Používají se tedy především pro získání zpětné vazby uživatele. Umožňují získat jeho názor na nějakou problematiku, popř. na samotný test/dotazník. Pomocí takové otázky lze také získat odpověď, která tvůrce dotazníku vůbec nemusela napadnout.

2.7 Alternativní nástroje pro online testování znalostí a vytváření dotazníků

V průběhu posledních let vzniklo mnoho služeb pro vytváření online dotazníků, anket a testů. V následujícím textu je popsáno několik významných představitelů těchto slu-

žeb a u každé z nich jsou představeny její výhody i nevýhody. Všechny služby a nástroje, které jsou zde zmíněny, jsou buď zcela zdarma nebo poskytují neplacené cenové plány.

U všech zmíněných služeb se testovací sady a jejich otázky vytvářejí pomocí klikání, což pro laika může být intuitivnější než otázky vytvářet pomocí speciálních konfiguračních souborů jako je tomu u programu SelfTest.

Zásadní nevýhodou většiny těchto služeb může být, že v případě jejich použití svěřujeme uživatelská data a výsledky testů do rukou třetích stran. I přesto, že služby ve svých zásadách ochrany osobních údajů tvrdí [15], že k datům uživatelů nedostane přístup nikdo jiný než tvůrce testu a že nebudou data prodávat žádným jiným službám či organizacím, nemáme to zajištěno nijak jinak, než jejich slibem. Další nevýhodou je, že pro vytvoření testu, ankety nebo dotazníku je nutné se u služby zaregistrovat a vytvořit si u nich uživatelský účet. I to může být pro zadavatele testu nepříjemné.

2.7.1 ClassMarker

- www.classmarker.com

ClassMarker je služba speciálně zaměřená na profesionální vytváření testů a analýzu získaných výsledků. Služba nabízí bezplatný cenový plán, který poskytuje plnou funkcionalitu služby s jediným omezením, které představuje vytvoření maximálně sta testů měsíčně. Další nevýhodou je, že si služba v bezplatném plánu vyhrazuje právo na zobrazování reklam v testech [1].

2.7.2 SurveyMonkey

- www.surveymonkey.com

Jedná se o jednu z nejznámějších a nejpopulárnějších služeb pro vytváření online dotazníků [3, 4]. V neplacené verzi se ale hodí pouze pro vytváření menších dotazníků o maximálně deseti otázkách, u kterých není očekávána velká účast respondentů. V základní neplacené verzi je totiž možné získat výsledky pouze od sta uživatelů a není možné je jakýmkoliv způsobem exportovat. Při nutnosti testovat více uživatelů je tak nutné si zaplatit placenou verzi.

Výhodou naopak může být, že i v základní verzi podporuje až patnáct typů otázek [14]. Dále podporuje našeptávání při vytváření otázek (pouze v anglickém jazyce) nebo zobrazení výsledků v reálném čase.

2.7.3 Survio

- www.survio.cz

Představuje českou alternativu systému pro vytváření online dotazníků. Tato služba opět poskytuje neplacenou verzi, ve které nabízí vytvoření až pěti dotazníků. Pro každý z dotazníků je pak možné získat výsledky pouze od sta uživatelů [16].

Stejně jako SurveyMonkey (viz 2.7.2) v základní verzi neposkytuje export výsledků do nějakého lépe strojově zpracovatelného formátu. Jediný způsob získání výsledků co služba v základní verzi aktuálně poskytuje, je export do formátu PDF[8]. Výhodou je zde dostupnost

[8] *PDF (Portable Document Format)* - souborový formát pro ukládání dokumentů nezávisle na softwaru i hardwaru.

jak české online podpory, tak i české administrace včetně českého rozhraní systému vytváření dotazníků.

2.7.4 Moodle™

- www.moodle.org

Jedná se o robustní elektronický výukový systém, který je poskytován zdarma jako *Open-Source* pod svobodnou licencí GPL. Tento systém je hojně využíván školami a univerzitami ke vzdělávání po celém světě. Výhodou je tak jeho ověřenost studenty napříč celým světem a dostupnost ve více jak sto jazycích [6].

Vytváření testů a následné testování je pouze jednou z mnoha funkcí, kterými tento systém disponuje.

Systém Moodle™ je zdarma volně ke stažení[9], ale bohužel vyžaduje značné technické znalosti při jeho nasazování a následné konfiguraci [4]. To může být pro málo technicky zdatné uživatele problém. Proto se určitě nehodí pro vytvoření jednorázového testu. Výhodou systému je podpora pro velké množství různých typů testových otázek [7]. Průběh testu lze velmi detailně konfigurovat. Je možné nastavit věci jako náhodné prohazování pořadí odpovědí, prohazování pořadí otázek, počet pokusů ke zvládnutí testu, různé druhy časování a mnoho dalších věcí [8].

[9] Viz `http://download.moodle.org`

Kapitola 3

Návrh rozšíření

Kapitola popisuje návrh různých rozšíření programu SelfTest. V první části kapitola představuje různé statistiky otázek jako stabilitu otázky, úspěšnost zodpovězení otázky či průměrnou dobu potřebnou pro zodpovězení otázky. Dále jsou zde navrženy skripty pro vyhodnocování parametrů testovací sady a pro zobrazování výsledků uživatelů. Nemalou část kapitoly tvoří návrh metrik pro analýzu testovací sady. Příkladem mohou být metriky počítající pokrytí sady otázek tématickými okruhy, popř typem otázek.

3.1 Požadavky na statistiky otázek a odpovědí

Cílem následujících analýz je získání statistik charakterizujících otázky v testovací sadě především na základě již získaných uživatelských odpovědí. Jednotlivé analýzy pracují s daty jako je doba uživatelovi odpovědi, výsledek odpovědi (jestli byla správná nebo špatná) nebo s celkovou úspěšností účastníka v testu.

3.1.1 Průměrná doba potřebná pro zodpovězení otázky

Průměrná doba potřebná pro zodpovězení otázky q je dána aritmetickým průměrem:

- $Atime_{q_i}$ – doba trvání odpovědi
- $Rcount_q$ – počet všech odpovědí na otázku

$$averageTime_q = \frac{\sum_{i=1}^{Rcount_q} Atime_{q_i}}{Rcount_q} \tag{3.1}$$

Tato hodnota znázorňuje, jak dlouho průměrně trvá účastníkům testu zodpovědět otázku q.

3.1.2 Úspěšnost zodpovězení otázky

Úspěšnost U zodpovězení otázky udává, kolik procent odpovědí na otázku bylo správných a kolik špatných.

Je dána poměrem:

- C_r – počet správných odpovědí na otázku
- C – počet všech odpovědí na otázku

$$U = \frac{C_r}{C} \times 100\% \tag{3.2}$$

Nízké hodnoty úspěšnosti nemusí nutně představovat obtížnost otázky - otázka nemusí být těžká na zodpovězení, může být pouze špatně položená (viz stabilita otázky 3.1.3). Dále je nutno zdůraznit, že jako špatně zodpovězená otázka se bere i ta, u které uživateli vypršel časový limit na její odpověď.

3.1.3 Stabilita otázky

Stabilita otázky vyjadřuje, do jaké míry se otázka odlišuje od ostatních otázek. Detekuje především otázky, kde budou uživatelé nuceni hádat odpověď. Uživatelé jsou nuceni hádat u otázek, pro které neznají odpověď. Tyto otázky jsou buď nepřiměřeně těžké nebo mají špatně položené zadání.

Stabilita je počítána na základě odpovědí účastníků testu. Čím více výsledků pro danou otázku máme, tím bude hodnota stability přesnější. Stabilita nabývá hodnot v těchto intervalech:

Interval [%]	**Popis**
$(0, 50\rangle$	Otázka je spíše nestabilní
$(50, 100\rangle$	Otázka je spíše stabilní

Příklad stoprocentní stability

Pokud budeme mít deset uživatelů, kteří se na test připravovali, měla by většina z nich (nejlépe všichni) zodpovědět všechny otázky správně. Pokud budeme mít dalších deset uživatelů, kteří se na test **neučili**, měla by se jejich úspěšnost v testu blížit k nule - nejlépe by všechny otázky měli mít zodpovězené **špatně**. Pokud bychom si o otázkách následně nechali vypsat statistiky, byla by jejich jejich úspěšnost přibližně 50%. Stabilita by ale byla **stoprocentní**, protože všichni, kteří odpověď měli znát ji znali a ti, kteří ji neměli znát, ji neznali.

Nulová stabilita

Aby stabilita otázky q mohla dosáhnout hodnoty 0%, museli by všichni uživatelé, kteří se na test nepřipravovali, dosáhnout v testu nulové úspěšnosti a zároveň otázku q zodpovědět správně. Dále by všichni uživatelé, kteří se na test připravovali, museli v testu dosáhnout stoprocentní úspěšnosti a otázku q zodpovědět chybně. Oba zmíněné stavy jsou už z principu nemožné. Stabilita o hodnotě 0% tedy nemůže nikdy nastat.

Příklad nízké stability

Mějme test o deseti otázkách. Pokud budeme mít deset uživatelů, kteří 9 otázek zodpověděli správně a desátou otázku q nikoliv a zároveň dalších deset uživatelů, kteří 9 otázek měli **špatně** a pouze desátou otázku q správně, pak otázka q bude velmi nestabilní. Její stabilita se bude limitně blížit k nule (v tomto konkrétním případě bude mít hodnotu 14%).

3.2 Vyhodnocování parametrů testovací sady

Program SelfTest nedisponoval žádnými prostředky, kterými by bylo možné efektivně měnit konfigurace otázek v testovací sadě, která již byla ostře spuštěna. Takovou situaci by mělo být možné vyřešit následujícími nástroji, které se zabývají především znovuvyhodnocováním již existujících výsledků.

3.2.1 Znovuvyhodnocení všech odpovědí pro danou otázku

Vstupem skriptu je identifikátor otázky. Pro tuto otázku skript v databázi nalezne všechny odpovědi účastníků. Znovu je vyhodnotí nad tou stejnou (aktualizovanou) případně jinou (explicitně zadanou) otázkou. Bude sloužit k ladění otázek a kontrole validity odpovědí především u otázek, kde se správnost odpovědi ověřuje regulárním výrazem (viz otázky typu *„grep“* 2.6.2).

3.2.2 Znovuvyhodnocení odpovědí účastníka testu nad testovací sadou

Skript na svém vstupu očekává identifikátor účastníka testu. V případě SelfTestu je tímto identifikátorem uživatelův e-mail. Pro tohoto uživatele v databázi nalezne všechny historické odpovědi. Ty následně znovu vyhodnotí nad aktuální, popř. jinou (explicitně zadanou) testovací sadou otázek a zobrazí, jak by daný účastník dopadl. Zobrazí tedy, které otázky by nově zodpověděl správně, které špatně, a celkovou úspěšnost v testu. Otázky typu *„noscore“* (viz podkapitola 2.6.2) vyhodnocovány nejsou. Otázky, u kterých uživateli vypršel čas na odpověď, jsou opět vyhodnoceny jako **chybně zodpovězené**.

3.3 Metriky testovací sady

V této části práce bude popsáno několik navržených metrik testovací sady. Metriky testovací sady popisují její charakteristiky, jako například počet otázek v testovací sadě, počet otázek, které jsou nabídnuty při testu uživateli, maximální délku testu, obtížnost otázek, úroveň kvality testovací sady apod. Některé z nich budou v rámci této práce rozebrány podrobněji a následně bude popsána jejich implementace.

Kategorizace otázek

Otázky bylo dříve možné rozlišovat pouze na úrovni oblastí (viz podkapitola 2.6.1), u kterých může každá otázka patřit vždy pouze do **jedné** oblasti. To je ale pro potřebné analýzy nedostatečné. Chceme mít možnost otázku zařadit detailněji. K tomu slouží implementace tzv. *tématických okruhů*.

Aby bylo možné zařadit jednu otázku do libovolného počtu tématických okruhů, je nástroj rozšířen o možnost přidat do konfigurace otázek informaci o tématických odvětvích, do kterých daná otázka spadá.

3.3.1 Metrika pokrytí sady otázek tématickými okruhy

Výstupem této metriky je zobrazení pokrytí testu tématickými okruhy (viz podkapitola 3.3). Pokrytí sady otázek tématickými okruhy říká, kolik otázek uživatel průměrně na dané téma dostane.

Pokrytí je dáno následujícím vzorcem:

- C – množina všech tématických okruhů (categories)
- A – množina všech oblastí (areas)
- Q – množina všech otázek v testovací sadě (questions)
- Q_a – množina všech otázek z oblasti $a \in A$, kde $Q_a \subseteq Q$
- Q_c – množina všech otázek spadajících do tématického okruhu $c \in C$, kde $Q_c \subseteq Q$
- a_{qtt} – *„Questions to test“* - počet otázek, který se z oblasti $a \in A$ náhodně vybere

$$coverage_c = \sum_{a \in A} |Q_a \cap Q_c| \frac{a_{qtt}}{|Q_a|} \tag{3.3}$$

3.3.2 Metrika zastoupení typů otázek v sadě

Tato metrika testovací sady zkoumá, v kolika případech dostane uživatel průměrně v odpovědi na výběr (uzavřená otázka), v kolika případech bude muset něco vyplnit (otevřená otázka) a v kolika případech bude muset napsat kus kódu, popř. něco vyřešit.

3.3.3 Metrika počtu otázek vybíraných z oblasti

Testovací sada může ve svých oblastech obsahovat daleko více otázek než je v průběhu testu účastníkovi nabídnuto. Tato metrika sleduje, kolik otázek z oblasti musí účastník zodpovědět vůči tomu, kolik jich oblast reálně obsahuje.

Jejím výstupem je informace ve formátu $area_i$: $X/C => Y\%$, kde $area_i$ představuje určitou oblast i, X je počet vybíraných otázek z oblasti (v názvosloví programu SelfTest jde o tzv. proměnnou `question_to_test`), C je celkový počet otázek v oblasti a proměnná Y tento poměr vyjadřuje procentuálně.

3.3.4 Metrika maximální doby trvání testu

Metrika sleduje maximální možnou dobu, která může být nutná pro průchod testem, kdyby uživatel vždy odpovídal na poslední chvíli a z oblasti by mu byly vždy vybrány otázky, které mají nejdelší možný čas pro odpověď.

Výstupem výpočtu je maximální doba, popř. výpis otázek, kterými by uživatel musel v testu projít, aby této maximální doby dosáhl. V případě, že se v testu vyskytne otázka, která nemá omezenu dobu odpovědi, je výsledná doba rovna speciální hodnotě -1. Ta říká, že uživateli může být nabídnut takový sled otázek, ze kterého nebude jasné, kdy test přesně skončí.

3.4 Zobrazení podrobných informací o výsledcích testu

SelfTest do dnešní doby podporoval pouze zobrazení správných a špatných odpovědí a konečnou úspěšnost v testu. Díky nové implementaci tématických okruhů, do kterých můžeme zařazovat jednotlivé otázky (viz podkapitola 3.3), lze konečný výsledek uživatele analyzovat podrobněji. Je například možné zobrazit, jak uživatel v jednotlivých okruzích obstál.

3.4.1 Úspěšnost účastníka testu v tématických okruzích

Analyzujeme z kolika procent účastník splnil jednotlivé tématické okruhy testovací sady. Už zde nerozlišujeme pouze které otázky uživatel splnil a které ne, ale zkoumáme i o jakých odvětvích má větší přehled, a o kterých naopak ví pouze okrajově.

3.4.2 Úspěšnost všech účastníků v tématických okruzích

Znázorňuje, jak si **všichni** uživatelé stojí v jednotlivých tématických okruzích. Je tak například možné zjistit, že většina uživatelů nemá dostatečné znalosti nějakého určitého okruhu.

Kapitola 4

Implementace rozšíření

Implementace navržených řešení byla rozdělena do několika skriptů podle svého účelu. Tato kapitola jednotlivé skripty podrobněji rozebírá a ukazuje jejich rozhraní. Ukazuje jaké nástroje byly použity pro generování grafů a jaký modul byl vytvořen pro tisk výsledných statistik na standardní výstup. Dále jsou v této kapitole detailněji popsaný implementované algoritmy.

V průběhu vývoje byla snaha co nejvíce funkcí pro výpočty integrovat přímo do rozhraní jednotlivých tříd modulů samotného programu. V nástrojích se pak těchto rozhraní využívalo a nebylo tak nutné mít všechnu funkcionalitu implementovánu přímo v daném skriptu. Díky tomu je možné k počítaným statistikám jednoduše přistupovat i v rámci samotného programu.

Výsledné nástroje pracují především v prostředí terminálu. Vypočtené hodnoty jsou vypisovány na standardní výstup. Zároveň je možné některé z hodnot vykreslovat do grafu.

4.1 Zobrazování statistik o otázkách

Všechna funkcionalita pro zobrazování statistik o otázkách byla implementována ve skriptu `print-question-stats.py`. Tento skript pro své řízení využívá třídu `QStatsManager`, která je implementována ve mnou navrženém modulu `qstats`.

Řízení skriptu se provádí pomocí přepínačů a jejich argumentů. Skriptu je nutné poskytnout alespoň jeden povinný argument, kterým je cesta. Tato cesta může být zadána jako celá cesta k otázce (například `seb-test/lehke/2`). Pak se statistiky počítají pouze pro danou otázku. Nebo může být zadána pouze částečně (jako cesta k oblasti nebo testovací sadě). Skript pak statistiky počítá pro všechny otázky v dané oblasti, popř. testovací sadě. To, jaké statistiky se budou vypisovat, je závislé na zadaném přepínači. Po dokončení výpočtu jsou statistiky vypsané na standardní výstup nebo, pokud je zadán přepínač „`-p`“, vykreslovány do grafu.

Skript podporuje všechny přepínače popsané v následující tabulce:

Přepínač	**Popis**
`-h, --help`	Vypíše kompletní nápovědu programu.
`-r, --success-rate`	Vypíše úspěšnost otázky, popř. skupiny otázek.
`-d, --answer-time`	Vypíše průměrný čas odpovědi zadané otázky.
`-s, --stability`	Vypíše stabilitu otázky.
`-p, --plot`	Vykreslí hodnoty do grafu.
`-a, --all`	Vypíše všechny výše zmíněné statistiky najednou.

Pro účely textového výpisu byl vytvořen modul `tableprint`, který má za úkol požadovaná data přehledně odsadit a vypsat do tabulky. Je založen na již testy ověřeném algoritmu [2].

Ukázku jeho použití lze vidět na příkladu kódu č. 4.1. Jeho použití je velmi intuitivní. Nejprve se vytvoří instance třídy `TablePrint` (1), která představuje tabulku. Metoda `addColumn()` (2) přidává do tabulky nové sloupce, `addRow()` (3) přidává do tabulky jednotlivé řádky. Tabulku je možné vytisknout vestavěnou funkcí jazyka Python - `print` (4).

```
1  from tableprint import TablePrint
2  table = TablePrint()            # (1)
3
4  table.addColumn("Uživatel")     # (2)
5  table.addColumn("Úspěšnost")
6
7  for uzivatel in uzivatele:
8      table.addRow(uzivatel, uzivatel.uspesnost) # (3)
9
10 print table                     # (4)
```

Kód 4.1: Ukázka použití modulu TablePrint.

Pro vykreslování grafů bylo využito knihovny *matplotlib* (viz podkapitola 2.2.1). Krátký příklad jejího použití pro vykreslení grafu funkce *sinus* lze vidět na ukázce kódu č. 4.2.

```
1  import math
2  import matplotlib.pyplot as plot
3
4  xs = [0.01*x for x in range(1000)] # (1)
5  ys = [math.sin(x) for x in xs]     # (2)
6  plot.plot(xs, ys)                  # (3)
7  plot.show()                        # (4)
```

Kód 4.2: Ukázka vykreslení grafu pomocí knihovny matplotlib.

Nejdříve vytvoříme souřadnice pro osu X od 0 do 10 v kroku 0.01 (1). K nim pak dopočítáme jejich hodnoty pomocí matematické funkce *sinus* (2). Seznamy souřadnic X a Y předáme knihovně (3) a následně je zobrazíme (4).

4.1.1 Algoritmus výpočtu úspěšnosti odpovědí na otázku

Algoritmus výpočtu úspěšnosti je implementován přímo ve třídě představující otázku `stQuestion` ve funkci `get_avg_success()`. Algoritmus prostřednictvím modulu `results` získá všechny odpovědi pro danou otázku. Tyto odpovědi následně postupně prochází a počítá kolik z nich bylo zodpovězeno správně. Pak už pouze pomocí vzorce (3.2) spočítá výslednou úspěšnost otázky. Otázky typu *„noscore“* (viz podkapitola 2.6.2) jsou při výpočtu ignorovány a namísto výsledné úspěšnosti je navrácena hodnota `None`.

V případě, že pro danou otázku nebyly nalezeny žádné odpovědi, je výsledná úspěšnost rovna hodnotě -1.

4.1.2 Algoritmus výpočtu stability otázky

Implementaci algoritmu pro výpočet stability lze opět nalézt ve třídě otázky `stQuestion`, konkrétně ve funkci `get_stability()`. Otázky typu *„noscore“* jsou při výpočtu opět přeskočeny a stabilita se u nich už z jejich podstaty nepočítá.

Pseudokód výpočtu stability

Pseudokód výpočtu stability lze nalézt na obrázku č. 4.3.

```
def vrat_stabilitu_otazky(otazka) {
    odpovedi = otazka.vrat_vsechny_odpovedi()

    pom = 0.0
    for odpoved in odpovedi:
        if odpoved.je_spravna():
            pom += CELKOVA_Uspesnost_ucastnika_teto_odpovedi_v_TESTU - 0.5
        else:
            pom += 0.5 - CELKOVA_Uspesnost_ucastnika_teto_odpovedi_v_TESTU

    pom /= odpovedi.pocet

    # Vyjadreni stability v procentech:
    stabilita_v_procentech = (pom + 0.5) * 100
    return stabilita_v_procentech
```

Kód 4.3: Pseudokód výpočtu stability otázky.

Vytvoříme si pomocnou proměnnou `pom`, pomocí které budeme počítat hodnotu stability. Její počáteční hodnotu nastavíme na 0. Při výpočtu se procházejí všechny odpovědi na otázku `otazka`. Pro každou odpověď je potřeba získat jejího autora a jeho celkovou úspěšnost v testu v procentuálním vyjádření. Jelikož SelfTest celkový výsledek testu uživatele nikde neukládá, je nutné si jej znovu spočítat. Na to už třída `stUserResults` modulu `results` poskytuje funkci `get_success()`.

Jakmile je spočtena úspěšnost uživatele, můžeme počítat stabilitu. Pokud je daná odpověď správná, přičteme k pomocné proměnné úspěšnost účastníka zmenšenou o hodnotu 0.5 (neboli 50%). Pokud je odpověď špatná, zvětšíme pomocnou proměnnou o 0.5 zmenšených o úspěšnost účastníka. Na konci algoritmu pomocnou proměnnou podělíme počtem všech odpovědí. Díky tomu získáme hodnotu v intervalu $(-0.5, 0.5\rangle$. Pro převedení hodnoty na procenta posuneme její interval o 0.5 doprava a vynásobíme hodnotou 100. Získáme tak konečnou hodnotu stability, kterou vrátíme.

4.1.3 Algoritmus výpočtu průměrné doby zodpovězení otázky

Algoritmus je, stejně jako předchozí dva, založený na procházení všech odpovědí dané otázky. Z těchto odpovědí čte dobu, kterou se odpovídající uživatel rozhodoval, než odpověděl. Takto získané časy nakonec zprůměruje podle vzorce 3.1) a vrátí výsledek.

4.2 Zobrazování statistik o testovací sadě

Statistiky a informace o testovací sadě vypisuje skript `print-testset-stats` na standardní výstup - stejně jako je tomu v případě vypisování statistik o otázkách pomocí modulu `tableprint` (viz podkapitola 4.1). Grafy jsou vykreslovány pomocí knihovny *matplotlib* (viz podkapitola 2.2.1). Jako nepovinný argument je při spuštění očekáván výčet jmen testovacích sad, pro které mají být statistiky vypsány. Skript lze ale spustit i bez tohoto výčtu argumentů. V takovém případě vypisuje statistiky pro všechny testovací sady, které jsou v adresářové struktuře aktuálně k dispozici.

Jeho rozhraní podporuje tyto přepínače:

Přepínač	Popis
`-h, --help`	Vypíše kompletní nápovědu programu.
`-c, --coverage`	Vypíše hodnoty pokrytí tématickými okruhy otázek.
`-t, --type-coverage`	Vypíše hodnoty pokrytí typem otázek.
`-p, --plot`	Vykreslí hodnoty do grafu.

4.2.1 Zobrazení pokrytí tématickými okruhy

Algoritmus pro výpočet pokrytí je implementován v modulu `questions` v rámci třídy `stTestSet` a její funkce `get_coverage()`.

Datovou strukturou, do které si algoritmus zaznamenává jednotlivá pokrytí, je slovník. Jeho klíčem je název okruhu a jeho hodnotou je jeho hodnota pokrytí. Algoritmus postupně prochází všechny otázky dané testovací sady. Pro každou z otázek získá její seznam okruhů. Pak už pouze provede výpočet pokrytí pro každý z těchto okruhů podle vzorce (3.3), které si interně uloží do výše zmíněného slovníku. Po tom, co algoritmus projde všechny otázky testovací sady, se ve slovníku budou nacházet konečné hodnoty pokrytí.

Příklad výpočtu pokrytí u dané testovací sady

Na obrázku č. 4.4 je v hranaté závorce u jména oblasti číslem uveden počet otázek, který z oblasti při testu náhodně vybíráme (proměnná Q_{tt} - hodnota *questions to test*). Jednotlivé odrážky představují otázky. Otázka má vždy své číslo a za dvojtečkou výčet okruhů, do kterých spadá.

```
1  lehke [2]:
2    - 1: kernel
3    - 2: kernel,tcp
4
5  tezsi [1]:
6    - 1: kernel
7    - 2: tcp
8
9  tezke [1]:
10   - 1: kernel
11   - 2: tcp
12   - 3: kernel,shell
```

Kód 4.4: Příklad testovací sady.

Zde tedy vybíráme dvě otázky z oblasti `lehke`, jednu z oblasti `tezsi` a jednu z oblasti `tezke`. Pro tuto testovací sadu chceme vypočítat její pokrytí tématickými okruhy. Vzorce (4.1) a (4.2) demonstrují výpočet pokrytí pro okruh *kernel*. Výpočet probíhá na základě vzorce (3.3).

$$kernel = \frac{Q_{tt_{lehke}}}{C_{lehke}} + \frac{Q_{tt_{lehke}}}{C_{lehke}} + \frac{Q_{tt_{tezsi}}}{C_{tezsi}} + \frac{Q_{tt_{tezke}}}{C_{tezke}} \tag{4.1}$$

$$kernel = \frac{2}{2} + \frac{2}{2} + \frac{1}{2} + \frac{1}{3} + \frac{1}{3} \doteq 3.17 \tag{4.2}$$

Příklad výstupu:

Příklad výstupu analýzy pokrytí této testovací sady jednotlivými okruhy lze vidět na obrázku č. 4.5.

```
Test-set: categories
--------------------
Category  Coverage
kernel    3.17
shell     0.33
tcp       1.83
```

Kód 4.5: Výstup pokrytí tématickými okruhy.

Tento výstup říká, že pokud uživatel projde celým testem, dostane průměrně 3.17 otázky z okruhu *kernel*, 0.33 otázky z okruhu *shell* a 1.83 otázky z okruhu *tcp*.

4.2.2 Zobrazení pokrytí testovací sady typem otázek

Výpočet pokrytí testovací sady typem otázek probíhá podobně jako při výpočtu pokrytí tématickými okruhy. Jediný rozdíl je v tom, že každá otázka je vždy **právě jednoho** určitého typu. V případě pokrytí tématickými okruhy může otázka náležet do N různých okruhů.

Příklad výstupu výpočtu pokrytí typem otázek u testovací sady `testset` lze vidět na ukázce č. 4.6.

```
> PYTHONPATH=src python2 tools/scripts/print-testset-stats.py -t testset
Test-set types coverage: testset
Question type  Coverage
grep           1.00
exec           1.00
abc            2.00
noscore        1.00
-------------------
Open-ended: 10.00 %
Close-ended: 80.00 %
```

Kód 4.6: Výstup pokrytí typem otázek.

Výstupem je výpis jednotlivých typů otázek v levém sloupci, v pravém jsou hodnoty pokrytí. Pokud uživatel projde touto testovací sadou, budou mu v průběhu testu průměrně nabídnuty dvě otázky typu *„abc“*, jedna typu *„grep“*, jedna typu *„exec“* a jedna typu *„noscore“*. Na konci skript vypíše procentuální zastoupení otevřených a uzavřených otázek.

4.3 Zobrazování výsledků uživatelů

Pro potřeby zobrazování výsledků uživatelů byl vytvořen skript `print-user-stats.py`. Tento skript je zaměřen především na vykreslování výsledků uživatelů, a to jak pro konkrétního uživatele (přepínač „`-u`“), tak i všech uživatelů jako celku. Bez zadaného přepínače program vypíše statistiky pro **všechny** uživatele v databázi. Pro svůj běh využívá modul `ustats`, ve kterém se nacházejí funkce pro výpis jednotlivých statistik. Získávání uživatelských výsledků je zajištěno modulem `results`. Tento modul byl speciálně upraven a rozšířen o funkce počítající úspěšnost - `get_success()` a `get_results_rate()`.

Jsou podporovány tyto přepínače:

Přepínač	Popis
`-h, --help`	Vypíše kompletní nápovědu programu.
`-u, --user`	Vypíše statistiky pouze pro konkrétního zadaného uživatele.
`-r, --results`	Zobrazí výsledky uživatele pro jednotlivé otázky. Nakonec vypíše uživatelovu konečnou úspěšnost v testu. Musí být zadán s přepínačem „`-u`“.

4.4 Znovutestování odpovědí

Pro účely znovutestování byl vytvořen skript `retest.py`, ve kterém je implementováno pouze zpracování přepínačů. Vlastní algoritmy pro znovutestování jsou implementovány ve třídě `RetestManager` v modulu `qstats`. Skript je schopný znovutestovat jak výsledky určitého uživatele testu, tak i všechny výsledky dané otázky. Zadaným přepínačem se rozhodne, jaké znovutestování se bude provádět.

Podporované přepínače jsou následující:

Přepínač	Popis
`-h, --help`	Vypíše kompletní nápovědu programu.
`-u, --user`	Spustí znovutestování výsledků zadaného uživatele.
`-q, --question`	Spustí znovutestování všech výsledků zadané otázky.
`-a, --all`	Vypíše i ty výsledky, které se od minulého testování nezměnily.
`-r, --retest`	Umožňuje explicitně změnit cílovou testovací sadu, na které se budou odpovědi testovat.
`-c, --update`	Zaktualizuje výsledky uživatele v databázi.

Skriptu musí být poskytnut jeden z přepínačů `--user` nebo `--question`. Každý z nich očekává i argument. V případě `--user` se jako argument očekává identifikátor (e-mailová adresa) uživatele jehož výsledky chceme testovat. U přepínače `--question` je jako argument očekáván identifikátor otázky.

Odpovědi jsou implicitně vyhodnocovány nad **původní** testovací sadou. To lze ovlivnit přepínačem `--retest`, díky kterému je toto chování možné změnit a prostřednictvím něho nastavit jinou sadu otázek, nad kterou budou odpovědi vyhodnocovány.

4.4.1 Implementace algoritmů

Znovuvyhodnocení jak odpovědí určitého účastníka testu, tak všech odpovědí dané otázky, probíhá velmi podobně. Nejprve se získají pomocí již několikrát zmiňovaného modulu `results` všechny odpovědi, které chceme znovu otestovat. V dalším kroku algoritmu se odpovědi postupně prochází a znovu testují.

Algoritmus znovutestování výsledků uživatele

Z každé odpovědi je nejprve získán objekt otázky `stQuestion`, ke které odpověď patří. Pomocí tohoto objektu je načtena konfigurace otázky z testovací sady. Následně je možné provést testování. K tomu třída `stQuestion` poskytuje metodu `evaluate_answer()`, která jako parametr přijímá odpověď. Tuto odpověď vyhodnotí nad otázkou a vrátí nový výsledek. Nakonec už pouze dochází k porovnání starého výsledku s výsledkem nově získaným.

Algoritmus znovutestování všech výsledků otázky

Rozdíl mezi tímto a předchozím algoritmem je pouze ten, že objekt otázky `stQuestion` je vytvořen již **před** procházením výsledků. V předchozím algoritmu bylo nutné otázku vytvářet při každé iteraci cyklu s výsledky.

Kapitola 5

Testování a demonstrace použitelnosti metrik

Tato kapitola ukazuje, jak a na jakých datech byla implementovaná rozšíření testována. Dále představuje použití implementovaných řešení v praxi.

5.1 Testování algoritmů pro zobrazování statistik o testovací sadě

Testování těchto algoritmů probíhalo na několika speciálně vytvořených testovacích sadách otázek. Hodnoty statistik pro tyto sady byly nejdříve spočteny manuálně. Následně bylo ověřeno, že automaticky spočtené hodnoty jsou správné. Testovací sady lze nalézt na přiloženém CD v adresáři `tests` (více viz příloha D).

5.2 Testování výpočetních algoritmů

Testování probíhalo jak na uměle vytvořených výsledcích, tak i na reálných datech. Umělá data byla využita pro ověření funkčnosti jednotlivých algoritmů při vývoji. Na reálných datech se pak testovala prospěšnost a použitelnost jednotlivých rozšíření.

5.2.1 Testování v průběhu implementace

V průběhu implementace byly výpočetní algoritmy testovány na umělých datech. Tyto data včetně testovací sady byla získána v rámci akce *Fedora 18 Release party* ve společnosti Red Hat a byla mi poskytnuta prostřednictvím pana Jana Hutaře.

5.2.2 Testování na reálných datech

Jelikož bylo potřeba otestovat algoritmy i na větším vzorku dat, byla pro další účely testování vytvořena nová testovací sada otázek, která byla zaměřena především na oblast informačních technologií, konkrétně oblast linuxového operačního systému. Její strukturu lze najít v příloze A. Aby mohl být test nabídnut co možná největšímu počtu uživatelů, bylo nutné jej spustit na veřejném serveru. Jako cílová platforma pro běh testu byl zvolen *OpenShift*, který je nástrojem SelfTest podporován. Následně byl test nabídnut pěti uživatelům, díky kterým se ještě doladila některá poslední nastavení otázek (jako například

doba odpovědi). Pak už byl test nabídnut cílové skupině uživatelů, jejíž velkou část tvořili především spolužáci. Tito lidé byli získáni prostřednictvím sociálních sítí a fór.

Testu se nakonec zúčastnilo více jak 112 lidí. Jednotliví účastníci testu byli většinou lidé z oboru informačních technologií. Ne všichni ale měli dobré linuxové znalosti. Každá otázka měla jednu správnou odpověď. V případě nezodpovězení nebo vypršení času odpovědi byla otázka vyhodnocena jako chybná. Díky tomu byli uživatelé nuceni tipovat u otázek, na které neznali odpověď. Dále uživatelé nesměli odpovědi nikde vyhledávat. To bylo částečně zajištěno právě omezením času odpovědi.

Součástí testu byla i jedna nesmyslně zadaná otázka, u které bylo nutné odpověď tipnout. Bylo očekáváno, že hodnota stability této otázky bude nízká (nižší jak 50%) a že otázka tedy bude nestabilní. To se nakonec potvrdilo.

Statistiky otázek na základě výsledků uživatelů

Nad získanými výsledky byl spuštěn skript pro získání statistik o otázkách testovací sady. Jeho výstup lze vidět na obrázku č. 5.1.

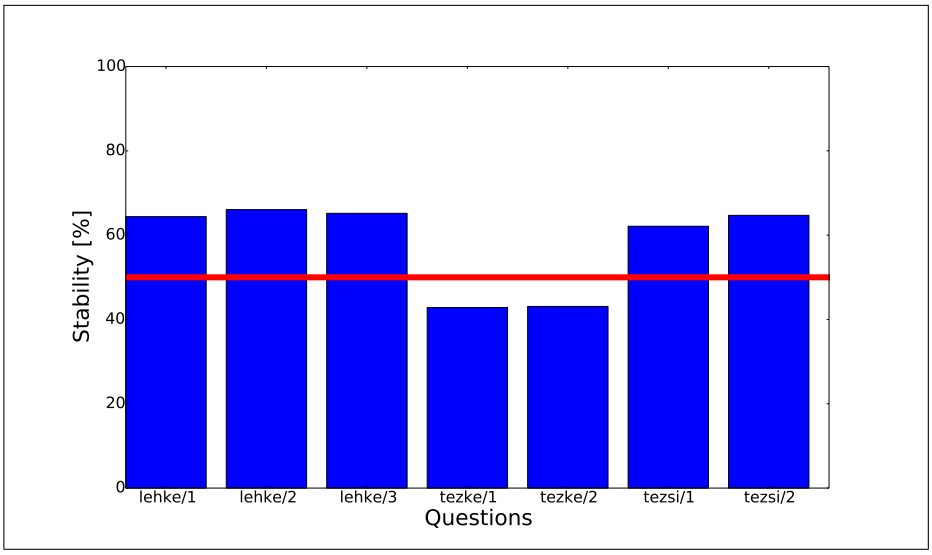
```
> PYTHONPATH=src python2 ./tools/scripts/print-question-stats.py --all seb-test
Test-set: seb-test
------------------
Question  Stability [%]    Avg. time [s]    Success [%]
lehke/1   64.39             9.32             67.94
lehke/2   66.06             7.41             93.08
lehke/3   65.18             8.46             94.62
tezke/1   42.84             25.46            10.77
tezke/2   43.09             15.39            11.63
tezsi/1   62.11             11.91            67.44
tezsi/2   64.70             7.37             96.92
```

Kód 5.1: Výstup skriptu pro výpis statistik o otázkách.

Na jednotlivých řádcích je vždy v prvním levém sloupci uveden identifikátor otázky. V dalších sloupcích jsou pak uvedeny jednotlivé statistiky. Z výstupu je patrné, že stabilita otázek `tezke/1` a `tezke/2` je nízká. Po další analýze výsledků se ukázalo, že otázka `tezke/1` byla nepřiměřeně těžká vzhledem ke znalostem účastníků testu. Otázka `tezke/2` byla cíleně zadána nesmyslně. Hodnoty stability znázorněné v grafu lze najít níže na obrázku č. 5.1. Červená čára udává hranici 50%, kde se stabilní otázky nacházejí nad ní.

Při testování byl také zjištěn problém s náročností výpočtu stability. Její výpočet i na velmi malém počtu výsledků trval relativně dlouho (v řádu vteřin). Na testu s více uživateli (a tedy i více výsledky) skript dokonce havaroval. Později bylo zjištěno, že nástroj SelfTest zbytečně otevíral a následně neuzavíral některá databázová spojení. Tím při výpočtu vznikala velká režie a mnoho neuvolněných zdrojů systému. V případě, že skript narazil na databázový limit současně vytvořených databázových spojení, havaroval. Problém byl vyřešen implementací tzv. *perzistentního databázového spojení* a opravou míst, kde spojení s databází nebylo uzavíráno. Spojení s databází je v současné době otevíráno při příchozím požadavku. Uzavřeno je až po vygenerování odpovědi (ještě před jejím odesláním zpět na klienta).

I přes tyto problémy se stabilita počítá relativně dlouho. Na stroji s procesorem *Intel® Core™ 2 Duo T7300* se dvěma jádry o maximální frekvenci $2Ghz$ a s 64bitovým operačním systémem Archlinux (Linux verze 3.14) trvá výpočet pro testovací sadu *seb-test* s výsledky od 112 uživatelů přibližně šest vteřin (viz výstup příkazu `time` na obrázku č. 5.2).



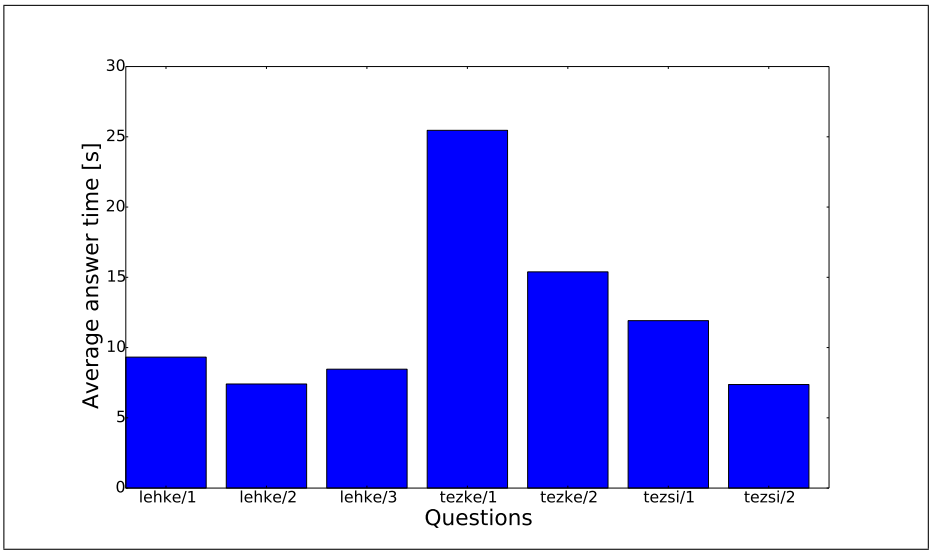
Obrázek 5.1: Znázornění stability otázek v grafu.

```
> time PYTHONPATH=src python2 tools/scripts/print-question-stats.py seb-test -s
real    0m5.860s
user    0m5.407s
sys     0m0.420s
```

Kód 5.2: Výstup programu time pro výpočet stability.

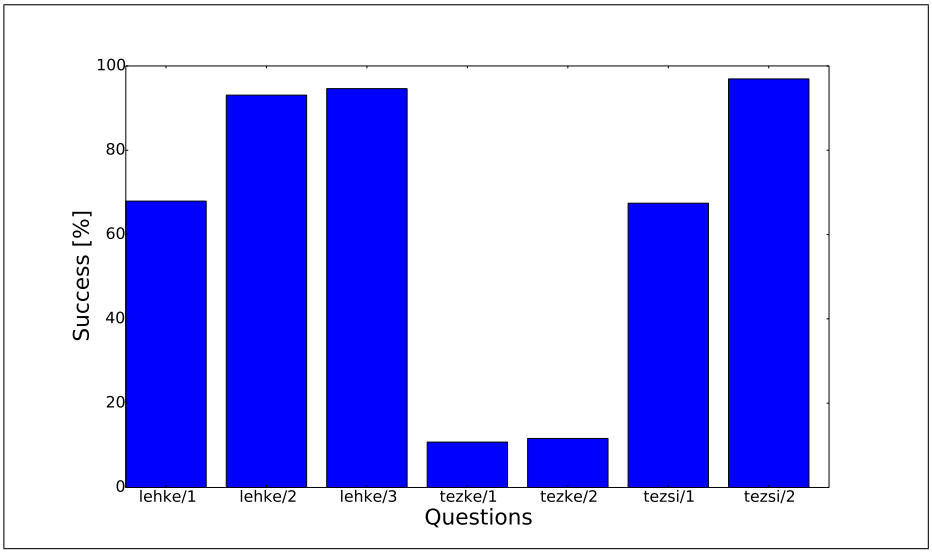
Výpočet stability by šlo dále optimalizovat. Bylo by možné si konečnou úspěšnost uživatele v testu ukládat ihned po tom, co test dokončí. Tím pádem by nebylo nutné ji znovu počítat.

V průběhu ověřování konfigurace testovací sady byl pro těžké otázky nastaven časový limit odpovědi na 30 vteřin. Dle lidí, kteří sadu ověřovali, byl tento limit příliš krátký a nestíhali odpovídat. Před nabídnutím testu cílové skupině byl tedy tento limit navýšen na 50 vteřin. V praxi se pak ukázalo (viz graf na obrázku č. 5.2), že 30 vteřin bylo pro těžké otázky plně dostačující.



Obrázek 5.2: Znázornění průměrné doby odpovědi pro jednotlivé otázek v grafu.

Jako poslední je zde na obrázku č. 5.3 znázorněn graf úspěšnosti uživatelů při zodpovídání jednotlivých otázek.



Obrázek 5.3: Znázornění úspěšnosti odpovědí na otázky v grafu.

Zobrazování výsledků uživatelů

Testování zobrazování výsledků nejdříve probíhalo na uměle vytvořených otázkách, ke kterým byly do databáze přidány umělé odpovědi. Po ověření správnosti generovaných výsledků byl algoritmus spuštěn nad databází odpovědí získaných prostřednictvím uživatelského testu *seb-test*. Na výstupu č. 5.3 lze v pravém sloupci vidět procentuální úspěšnost uživatelů v jednotlivých tématických okruzích testu.

Lze vypsat i procentuální úspěšnost v tématických okruzích pro konkrétního uživatele. To se provede pomocí přepínače `-u` jako je tomu na ukázce č. 5.4. Pokud by byla potřeba

spočítat jeho procentuální úspěšnost v testu a zobrazit jaké otázky zodpověděl správně a jaké špatně, lze použít přepínač `--results` (viz ukázka č. 5.5).

```
> PYTHONPATH=src python2 tools/scripts/print-user-stats.py
Category  Success [%]
windows   94.62
ssh       53.85
linux     58.02
net       53.85
os        67.80
bash      93.08
```

Kód 5.3: Úspěšnost účastníků testu seb-test v jednotlivých tématických okruzích.

```
> PYTHONPATH=src python2 tools/scripts/print-user-stats.py --user \
4e839fdb2eeaabc759e6a15a13a66cac@server.tld
Category  Success [%]
windows   100.00
ssh       50.00
linux     66.67
net       50.00
os        80.00
bash      100.00
```

Kód 5.4: Úspěšnost v tématických okruzích pro konkrétního uživatele.

```
> PYTHONPATH=src python2 tools/scripts/print-user-stats.py --results --user \
4e839fdb2eeaabc759e6a15a13a66cac@server.tld
seb-test/lehke/1: PASS
seb-test/lehke/3: PASS
seb-test/lehke/2: PASS
seb-test/tezke/2: FAIL
seb-test/tezke/1: FAIL
seb-test/tezsi/1: PASS
seb-test/tezsi/2: PASS
----------------------
Final success: 71.4285714286 %
```

Kód 5.5: Zobrazení celkové úspěšnosti uživatele v testu.

5.3 Testování skriptů pro znovutestování

Následující ukázky testování budou probíhat na testovací sadě, která bude obsahovat pouze jednu otázku typu *„grep“*.

Pro účely testování bude schválně regulární výraz, který ověřuje správnost odpovědi, zadán špatně. Ukázka její konfigurace je zobrazena na obrázku č. 5.6.

```
[question]
type=grep
question=Jaká písmena se v abecedě nacházejí mezi písmeny "A" a "D"?
correct=^\s*[bB]\s*$
```

Kód 5.6: Chybně zadaná otázka typu *„grep“*.

Mějme teď dva uživatele s identifikátory *b@srv.cz* a *c@srv.cz*. Každého z nich necháme projít testem. První z nich odpoví písmenem „B“ a druhý písmenem „C“. Chtěli bychom, aby obě odpovědi byly vyhodnoceny jako správné. To se ale nestane, protože regulární

výraz `^\s*[bB]\s*$` vyhodnocuje jako správná písmena pouze „b“ a „B“. Důkazem může být výstup skriptu `print-user-stats.py` na obrázku č. 5.7.

```
> PYTHONPATH=src python2 tools/scripts/print-user-stats.py -r -u b@srv.cz
d1/area/1: PASS
----------------------
Final success: 100.0 %

> PYTHONPATH=src python2 tools/scripts/print-user-stats.py -r -u c@srv.cz
d1/area/1: FAIL
----------------------
Final success: 0.0 %
```

Kód 5.7: Výstup statistik uživatele.

Regulární výraz tedy opravíme. Změníme ho na `^\s*[bBcC]\s*$`.

5.3.1 Testování znovuvyhodnocování všech odpovědí pro danou otázku

Následně je nutné ověřit, že oprava výrazu byla korektní. Za tímto účelem proto odpovědi nad nově upravenou otázkou znovu vyhodnotíme a necháme si vypsat změny ve výsledcích pomocí skriptu `retest.py`. Získáme výstup zobrazený na obrázku č. 5.8. Písmeno „P“ ve výstupu znamená, že odpověď regulárním výrazem prošla (otázka byla vyhodnocena kladně), „F“ znamená, že byla vyhodnocena negativně.

```
> PYTHONPATH=src python2 tools/scripts/retest.py -q d1/area/1
...
Correct: ^\s*[bBcC]\s*$
----------------------
User       Old state     New state     Change     Answer
c@srv.cz   F             P             +          C
```

Kód 5.8: Výstup znovutestování výsledků otázky.

Skript ve výchozím stavu vypisuje pouze výsledky, které se změnily. Z výstupu tedy vyplývá, že odpověď „C“, byla nově vyhodnocena jako správná.

5.3.2 Testování znovuvyhodnocování odpovědí účastníka testu

Regulární výraz byl opraven správně. Teď už pouze zbývá přepočítat úspěšnost účastníka *c@srv.cz* v testu. K tomu poslouží přepínač `--user`, kterému jako argument předáme identifikátor uživatele, u kterého chceme znovu otestovat jeho výsledky. Ve výstupu na obrázku č. 5.9 lze vidět, že odpověď uživatele již prošla bez problému.

```
> PYTHONPATH=src python2 tools/scripts/retest.py -u c@srv.cz
User: c@srv.cz
----------------------
Question              Old state     New state     Change
d1/area/1             F             P             +
----------------------
Score: [1/1] ~100 %
```

Kód 5.9: Výstup znovutestování výsledků uživatele.

Tento nový výsledek ale ještě není uložen v databázi. Aby se nový stav výsledků uživatele promítl i do databáze, je nutné tento skript spustit s parametrem `--update`.

Kapitola 6

Závěr

Hlavním cílem této práce bylo především vylepšit a rozšířit program SelfTest o nástroje, které umožňují analýzu jak testovací sady, tak i získaných výsledků od testovaných uživatelů. Za tímto účelem bylo implementováno několik skriptů, které tuto analýzu provádějí. Dále byl implementován skript, který provádí znovutestování výsledků uživatelů. V průběhu vývoje probíhalo refaktorování zbytečně složitého nebo nesrozumitelného kódu, ve kterém bylo nutné některá místa převést na objektový kód. Výsledná rozšíření byla testována jak na uměle vytvořených datech, tak i na reálných datech získaných prostřednictvím uskutečněného testu *seb-test*. Díky testování bylo nalezeno a opraveno mnoho chyb nejen v nově implementovaných částech programu, ale i v původním kódu.

Výsledkem je vydání nové verze nástroje - *SelfTest v2.0*[1]. Tato verze obsahuje všechny skripty pro analýzu a práci s výsledky uživatelů, které jsou v této práci popsané. Dále obsahuje všechna vylepšení a rozšíření, které bylo nutné z důvodu splnění zadání a požadavků firmy Red Hat implementovat jako: podpora SQLite, vylepšení systému pro tvorbu záznamů událostí a úprava rozhraní objektů. Všechna rozšíření programu byla testována jak manuálně, tak i pomocí automatických jednotkových testů. I nadále byl kladen důraz pro zachování podpory běhu na cloud computingové platformě OpenShift.

V průběhu vývoje bylo zařazeno přes 212 změn, ve kterých bylo upraveno více jak 71% celého kódu nástroje. Všechny zdrojové kódy včetně testovacích sad a databázových souborů s výsledky jsou k dispozici na přiloženém CD (více viz příloha D).

Tato práce mě naučila, že ne vždy píšeme kód nový, ale že v praxi je kód už většinou napsaný a je potřeba se v něm nejdříve vyznat a porozumět mu. Následně je až možné provádět změny a refaktorovat samotný kód. Dále mě tato práce nutila dodržovat zásady přispívání do projektu. Vytvářet menší řádně okomentované změny kódu, které se do repozitáře postupně zařazovaly. V neposlední řadě jsem se zdokonalil v jazyce *Python*. Naučil jsem se v jazyce orientovat a chápat některé jeho složitější konstrukce.

6.1 Další možný rozvoj nástroje

Před samotným rozšiřováním nástroje bych určitě doporučil ještě nějakou dobu kód testovat a dále refaktorovat. Refaktorováním myslím především nahrazení některých procedurálních konstrukcí, ve kterých se kód velmi často zbytečně opakuje, kódem objektovým. Následně by bylo možné SelfTest dále rozšiřovat.

[1] Viz https://git.fedorahosted.org/cgit/selftest.git/tag/?id=v2.0

Možná rozšíření

- **Podpora více typů otázek** – Přidat podporu jiných typů otázek. Např. chybí podpora pro otázky typu *ABC* s více možnými správnými odpověďmi.
- **Uživatelsky přívětivější chybová hlášení** – Zdokonalit chybové stránky a chybová hlášení zobrazovaná v průběhu testování tak, aby byly uživatelsky přívětivější a srozumitelnější.
- **Podpora vícejazyčnosti** – Upravit výstupy programu tak, aby bylo možné nástroj používat i v jiném než pouze anglickém jazyce.

Literatura

[1] ClassMarker: Privacy policy. [cit. 2014-04-28].
URL https://www.classmarker.com/online-testing/privacy/

[2] Ginstrom, R.: Pretty-printing a table in Python. 2007-09-04 [cit. 2014-05-09].
URL http://ginstrom.com/scribbles/2007/09/04/pretty-printing-a-table-in-python/#

[3] Henderson, T.: 4 Online Survey Tools: Which One is Best for You? 2012-01-31 [cit. 2014-04-26].
URL http://whinot.com/blog/2012/4-online-survey-tools-which-one-is-best-for-you/

[4] Leland, E.: A Few Good Online Survey Tools. 2011-02-10 [cit. 2014-04-26].
URL http://www.idealware.org/articles/fgt_online_surveys.php

[5] matplotlib development team: Python plotting. [cit. 2014-05-16].
URL http://matplotlib.org

[6] Moodle.com: Open-source learning platform. [cit. 2014-04-26].
URL https://moodle.org/

[7] MoodleDocs: Questions. [cit. 2014-04-28].
URL http://docs.moodle.org/26/en/Questions

[8] MoodleDocs: Quiz settings. [cit. 2014-05-06].
URL http://docs.moodle.org/23/en/Quiz_settings

[9] Pilgrim, M.: *Dive Into Python: Python from Novice to Pro*. CreateSpace, May 2012, ISBN 978-1475198119.

[10] Python Software Foundation: *WSGI Utilities and Reference Implementation*. [cit. 2014-04-28].
URL https://docs.python.org/2/library/wsgiref.html

[11] Robinson, D.; Coar, K.: *The Common Gateway Interface (CGI) Version 1.1*. Apache Software Foundation, October 2004, RFC 3875.
URL http://tools.ietf.org/html/rfc3875

[12] SQLite Consortium: About SQLite. [cit. 2014-04-19].
URL https://sqlite.org/about.html

[13] SQLite Consortium: Well-Known Users of SQLite. [cit. 2014-04-20].
URL https://sqlite.org/famous.html

[14] SurveyMonkey: Question Types Overview. [cit. 2014-04-25].
URL http://help.surveymonkey.com/articles/en_US/kb/What-types-of-questions-can-I-add-to-my-survey

[15] SurveyMonkey: Privacy policy. [cit. 2014-04-28].
URL https://www.surveymonkey.com/mp/policy/privacy-policy/

[16] Survio: Ceník prémiových služeb. [cit. 2014-04-26].
URL https://my.survio.com/orders/pricing

Příloha A

Struktura uživatelského testu

Test popsaný v následující části byl vytvořen za účelem testování různých analýz popsaných v této práci na reálných výsledcích uživatelů. Celý test se skládá ze sedmi otázek typu „abc“ (viz podkapitola 2.6.2) zaměřených především na operační systém GNU/Linux. Test byl spuštěn na serverech společnosti Red Hat na platformě *OpenShift* po dobu jednoho týdne. Během této doby byl nabídnut uživatelům.

A.1 Stromová struktura testovací sady

Testovací sadu otázek *seb-test* lze najít v adresáři `testsets` na přiloženém CD. Více o struktuře CD lze najít v příloze D. V následujícím textu je zobrazena a popsána adresářová struktura sady *seb-test* podrobněji:

```
testsets/
└── seb-test/
    ├── lehke/
    │   ├── 1/
    │   │   ├── config
    │   │   └── answers
    │   ├── 2/
    │   │   ├── config
    │   │   └── answers
    │   ├── 3/
    │   │   ├── config
    │   │   └── answers
    │   └── questions_to_test.txt ..... hodnota: 3
    ├── tezsi/
    │   ├── 1/
    │   │   ├── config
    │   │   └── answers
    │   ├── 2/
    │   │   ├── config
    │   │   └── answers
    │   └── questions_to_test.txt ..... hodnota: 2
    └── tezke/
        └── 1/
```

```
    │   ├── config
    │   └── answers
    ├── 2/
    │   ├── config
    │   └── answers
    └── questions_to_test.txt ..... hodnota: 2
├── badges
├── description
├── res/
│   └── img/
│       ├── bkg.png
│       └── logo.png
├── testset.config ..... konfigurační soubor testovací sady
└── welcome
```

A.2 Znění jednotlivých otázek

Otázky jsou v sadě rozděleny podle obtížnosti do tří oblastí - *lehké*, *těžší* a *těžké*. Každá oblast obsahuje soubor `questions_to_test.txt`. Hodnoty těchto souborů jsou zobrazeny ve výpisu struktury sady v podkapitole A.1. hodnoty udávají počet otázek, které se z dané oblasti nabídnou uživateli. Tyto otázky budou vždy náhodně zpřeházeny. Pro otázky z oblasti *lehké* byl nastaven limit času odpovědi na 20 vteřin, pro otázky z oblasti *těžší* byl nastaven na 30 vteřin. Pro otázky v poslední oblasti *těžké* byl limit nastaven na 50 vteřin. Správná odpověď je v následujícím výpisu otázek vždy zvýrazněna **tučně**.

Lehké

1. Kdo je tvůrcem operačního systému **GNU/Linux**?

   A) S. Jobs
   B) T. Linus
   C) M. Mackall
   D) A. Linux

2. Jakým příkazem na Linuxu vypíšete obsah **aktuálního adresáře**?

   A) wh
   B) where
   C) pwd
   D) ls

3. Jak se jmenuje jeden z prvních operačních systému od firmy Microsoft?

   A) Windows Vista
   B) Windows XP
   C) MS DOS
   D) Windows 1998
   E) Windows 7
   F) Windows 2000

Těžší

1. Jakou položku **ne**obsahuje soubor `/etc/passwd`?

   A) Shell
   B) Heslo
   C) Uživatelské jméno
   D) Cestu domovského adresáře

2. Jakým příkazem byste se připojili k serveru Merlin?

   A) connect merlin.fit.vutbr.cz
   B) ssh merlin.fit.vutbr.cz
   C) conn merlin.fit.vutbr.cz

Těžké

1. Jakým příkazem zkopírujete soubor `abc.txt` do adresáře `~/WWW` na server *seberm.com*, na kterém běží SSH na portu 3490?

   A) scp -P 3490 abc.txt seberm.com:~/WWW/
   B) scp abc.txt seberm.com:3490/~/WWW/
   C) scp abc.txt seberm.com:3490:~/WWW/
   D) scp -p 3490 abc.txt seberm.com/~/WWW/

2. Na jaké písmeno začíná slovo **tux**?

   A) 'x'
   B) 'y'
   C) 'z'

Otázka č. 2 sice byla zařazena do oblasti *těžké*, ale těžká nebyla. Byla zadána záměrně nesmyslně, aby na ní bylo možné testovat algoritmus pro výpočet stability otázky.

Příloha B

Funkčnost implementovaná nad rámec zadání

Aby byly nástroje popisované v práci plně funkční a aby byli lidé ze strany společnosti Red Hat s rozšířeními plně spokojeni, bylo nutné nástroj SelfTest rozšířit o další funkcionalitu. V následujících řádcích jsou zmíněna některá rozšíření, která byla implementována nad rámec zadání této práce.

B.1 Podpora SQLite3

V počáteční verzi program podporoval ukládání uživatelských výsledků a metadat pro testování pouze do souborů formátu CSV. Jelikož pro potřeby analýzy výsledků bylo potřeba provádět složitější dotazy, muselo být databázové rozhraní doplněno o další funkce, které jsou schopné vracet specifičtější výsledky a provádět složitější dotazy.

Práce s daty ve formátu CSV byla pro účely analýzy relativně pomalá, proto se hledalo alternativní řešení pro ukládání dat. Požadavkem bylo zachovat funkčnost i pro formát CSV. V rámci vylepšení bylo rozhodnuto, že bude SelfTest rozšířen o podporu databáze SQLite3.

Aby byla zachována podpora pro oba tyto typy databází, bylo nutné vytvořit tzv. *databázový adaptér*. Ten má za úkol rozhraní obou databází sjednotit. Program pomocí tohoto jednotného rozhraní může přistupovat k databázi, aniž by musel rozlišovat, jaký druh databáze používá.

Implementaci databázového adaptéru lze nalézt v souboru `database.py`. Rozhraní pro komunikaci s určitým typem databáze se nachází v souborech `databaseCSV.py` pro CSV a `databaseSQLite.py` pro SQLite3. O tom, která databáze bude pro práci s výsledky v programu využívána rozhoduje na základě hodnot v konfiguračním souboru `config.py` objekt `DbManager`.

B.2 Změna systému pro tvorbu záznamů událostí

Program SelfTest disponoval vlastním modulem pro záznam programových událostí. Tento modul byl zastaralý a bylo nutné jej refaktorovat. Jeho další nevýhodou byla malá škálovatelnost a potřeba dodatečných funkcí, které by bylo nutné v budoucnu implementovat.

V rámci rozšíření byl tento modul nahrazen modulem `logging`[1] ze standardní knihovny

[1] Viz https://docs.python.org/2/library/logging.html

jazyka Python. Tento nový modul pro záznam programových událostí předchozí modul plně zastupuje a navíc nabízí bohatší rozhraní pro různá nastavení.

Do programu přibyla podpora záznamu událostí do souboru, kterou je možné zapnout v hlavním konfiguračním souboru programu `config.py`. Toho bylo využito při testování na platformě *OpenShift*.

Příloha C

Ukázky konfiguračních souborů různých typů otázek

Skriptovatelná otázka „exec“

```
[question]
type=exec
question=<p>Napište příkaz pro vytištění obsahu aktuálního adresáře.</p>
correct=test
categories=bash,linux
```

Otázka s více odpověďmi „abc“

```
> FILE: config
[question]
type=abc
question=<p>Z následujících možností vyberte <strong>první</strong>.</p>
timeout=30
answers=@answers
correct=a
categories=others

> FILE: answers
a první
b druhá
```

Otázka typu „grep“

```
[question]
type=grep
question=Jaké písmeno patří mezi "B" a "D"?
correct=^\s*[cC]\s*$
default=prostě napište "C"
test_1=c
test_2=C
test_1_negative=Toto je nesprávná odpověď‘
categories=alphabet
```

Nehodnocená otevřená otázka „noscore“

```
[question]
type=noscore
question=<p>Tato otázka není hodnocena. Je vcelku jedno, co odpovíte.</p>
categories=others
```

Maticová otázka „matrix radio comment“

```
[question]
type=matrix_radio_comment
question=C|Bash|Python|Perl|Java
answers=vůbec neznám|viděl jsem|používám/používal jsem
comments=pokud jste danou technologii používal, popište jak a kde
timeout=-1
categories=programming
```

Příloha D

Obsah přiloženého CD

- `/`
  - `doc/` ..... zdrojové soubory práce včetně vygenerovaného PDF
  - `src/` ..... zdrojové soubory programu SelfTest
  - `testsets/` ..... sady otázek, na kterých byly algoritmy testovány
    - `d1/`
    - `d1-fixed/`
    - `seb-test/`
    - `cs-Fedora18/`
  - `results/` ..... databázové soubory s výsledky
    - `d1/`
      - `retesting.db` ..... data pro znovutestování uživatelských výsledků
    - `F18-data/`
      - `results.sql`
      - `users.sql`
    - `seb-test/`
      - `konecna_hashovana.db` ..... výsledky uživatelského dotazníku *seb-test*
  - `README.txt` ..... popis zprovoznění nástroje a pomocných skriptů
  - `AUTORI.txt` ..... autorství jednotlivých částí nástroje SelfTest
  - `LICENSE`