Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Přehled nástrojů pro automatické testování aplikací

Plzeň, 2014 Eduard Veselovský

Prohlášení

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

V Plzni dne 7.5. 2014 Eduard Veselovský

Abstract

This thesis focuses on tools for automatical software testing. The thesis describes the basic theoretical knowledge in the field of testing. The theory of testing is described as one of the parts of the software development process. In the theoretical section, a software error, and quality of software is defined, and automation testing fundamentals are mentioned. This part also contains a list of automated testing tools. The practical part describes in detail three selected tools and their fiction, and further it deals with sample cases of thein usage. In the conclusion, a survey was conducted if the sample cases were understandable.

Key words

Software testing, software bug, software development process models, automated testing, Test Studio, Jubula, Randoop.

Abstrakt

Tato práce se zaměřuje na nástroje pro automatické testování softwaru. Práce popisuje základní teoretické poznatky v oblasti testování. Teorie testování je popsána jako jedna ze součástí modelu vývoje softwaru. V teoretické části je definována softwarová chyba, kvalita softwaru, a popsány základy automatizace testování. Je zde i přehled automatizovaných nástrojů pro testování. Praktická část práce obsahuje bližší popis funkcí tří vybraných nástrojů, a dále se věnuje ukázkovým případům jejich použití. V závěru práce byl proveden průzkum, jsou-li ukázkové případy srozumitelné.

Klíčová slova

Testování softwaru, softwarová chyba, modely procesu vývoje softwaru, automatizované testování, Test Studio, Jubula, Randoop.

Obsah

1. Úvod

Pro toto téma jsem se rozhodl, protože jsem se chtěl seznámit s metodikou testování softwaru a její automatizací. Podle mého názoru je automatizace, alespoň částečná, tím správným krokem pro zlepšení kvality softwaru, protože díky automatizaci je možné testovat software vícekrát, s menšími náklady a rychleji než bez ní. Testování softwaru je jednou z důležitých částí vývoje softwaru, kterému je často věnována menší pozornost než v případě samotné implementace a kódování. Podceňování testování může mít neblahý vliv na kvalitu vyvíjeného softwaru, a tím i na spokojenost koncového zákazníka nebo uživatele.

Cílem práce je prozkoumat některé nástroje pro automatizaci testování. Abych se mohl pustit do automatizace testů a testování, musím nejdříve projít základy testování jeho metody a možnosti, a až poté se mohu zaměřit na jeho automatizaci. Proto část práce věnuji testování obecně a od základů, východiskem pro to, mi je kniha Testování Softwaru od Rona Pattona.

Pro nalezení nástrojů pro automatické testování musím projít trh s těmito nástroji, několik jich vybrat a vytvořit krátký přehled. Předně se budu věnovat nástrojům, které mohou testerovi poskytnout možnosti k vytváření testů automaticky, čímž myslím automatické generování testů na základě modelů, zdrojových textů, vytváření testů graficky nebo nahráváním za běhu vyvíjeného softwaru. Z nástrojů vyberu několik nejzajímavějších a ty prozkoumám podrobněji. V praktické části práce vytvořím ukázkové případy, které budou sloužit jako seznámení se s vybranými nástroji, ukázka práce s nimi a jejich užití. Následně předložím ukázkové případy co největšímu množství dobrovolníků, kteří ověří, zda jsou ukázkové případy srozumitelné. Práce neslouží k úplnému vysvětlení problematiky testování a jeho automatizace ani nemá prozkoumat celý trh s nástroji pro automatické testování. Spíše slouží k vytvoření přehledu některých zajímavých testovacích nástrojů a ukázkových případů.

2. Testování

Motivací pro testování je zajištění kvality v průběhu vývoje před uvedením produktu na trh nebo předáním produktu zákazníkovi. Testování softwarového produktu slouží pro ladění kódu a zajištění všech požadovaných funkcionalit. Kvalita testování závisí na kvalitě a preciznosti testovacího týmů. Testeři chyby hledají a vytváří jejich dokumentaci, ta je poté použita k odstranění chyby příslušným vývojářem. V samotném testování existuje i riziko nenalezení chyby, protože není možné otestovat software úplně a komplexně, většinou je produkt tak složitý, že nelze otestovat všechny možnosti, existuje velké množství vstupů a výstupů. Při vývoji softwaru většinou existuje i mezní termín pro vytvoření, tento termín může být zadán zákazníkem nebo je-li produkt na trh tak je mezní termín daný trhem. Ať už v případě produktu na trh, nebo zakázky od zákazníka je čas pro celý vývoj software předem daný a tím je daný i čas pro testování, tester musí vytvořit optimální množství testů, aby byl software dostatečně otestovaný, ale testy nezabíraly přílišné množství času a neprodlužovaly vývoj. Testy nikdy nedokazují, že chyby v produktu neexistují, někdy není úplně možné všechny chyby odstranit, a proto mají chyby svou prioritu, kterou určuje tester podle jeho vlastní úvahy. [1] [2]

Protože testování má zajišťovat kvalitu vytvořeného produktu existují i normy, ve kterých jsou popsány principy, charakteristiky a metriky pro stanovení kvality software, jednou z takových norem je norma ISO/IEC 9126 -1 která vymezuje šest charakteristik jakosti softwarového produktu. [3]

FUNKČNOST, jako schopnost produktu zabezpečit požadované funkce (důležité je, zda tyto funkce jsou zabezpečeny, nikoliv jak a za jakou cenu).

BEZPORUCHOVOST, jako schopnost produktu zajistit za daných podmínek požadovanou úroveň výkonu a poskytovaných služeb.

POUŽITELNOST, jako schopnost produktu být využíván při přiměřené míře úsilí potřebného na seznámení se s jeho možnostmi a jeho běžné provozování v daných podmínkách.

ÚČINNOST, jako schopnost produktu zajistit služby s přiměřenými nároky na zdroje systému a v přiměřené době.

UDRŽOVATELNOST, jako schopnost produktu být v průběhu používání měněn s cílem přizpůsobení požadavkům uživatele, odstranění zjištěných nedostatků, rozvoje a zlepšování funkcí nebo změny prostředí, ve kterém má pracovat (hardwarového, softwarového, ale i například legislativního).

PŘENOSITELNOST, jako schopnost produktu spolupracovat na datové i procesní úrovni s jinými systémy, včetně těch, které pracují na jiných platformách (datových, softwarových, ale i hardwarových).

Takovéto charakteristiky nejsou závazné a ne každý produkt je podle nich vytvořen, takovouto normu nejčastěji využívají velké společnosti jako Microsoft, Apple, Oracle. Příkladem, kdy se normy nepoužívají, je software vytvořený pro vlastní potřebu nebo od společností, které se přímo nezaměřují na vytváření software. [4]

2.1. Chyba

Chyba v software nebo také softwarová chyba je označení pro situaci, kdy software nebo část jeho kódu selhává v tom, co má dělat. Takovéto selhání můžeme označit několika výrazy: vada, závada, problém, omyl, událost, anomálie, odchylka, selhání, nekonzistence, chyba, bug. [1] Pojmů je mnoho a ne všechny znamenají vždy to stejné, rozdíly mohou být ve významu selhání nebo v závažnosti situace. Stejné selhání můžeme nazvat jinak, a tím mu dát jinou váhu například rozdíl mezi pojmy závada a událost, oba z těchto pojmů mohou stejnému selhání dát úplně jinou váhu. [1] Uvedu zde několik pojmů a jejich vysvětlení: [5]

- **Chyba**: Vyjadřuje nesoulad mezi hodnotou skutečnou a očekávanou například při výpočtu.
- **Selhání**: Vyjadřuje neschopnost systému provádět požadované operace či funkce.
- **Bug:** Je chyba v programu, kdy se program začne chovat neočekávaným způsobem
- **Porucha**: Označuje nesprávný krok, proces nebo špatná data v programu, který se díky tomu začne chovat neočekávaně.
- **Defekt**: Obyčejně poukazuje na několik problémů s chováním, vnějším či vnitřním, softwarového produktu.

Pro definici chyby v rámci testování software jsem vybral popis z knihy Testování software od Rona Pattona, který chybu nebo také softwarovou chybu popisuje pěti pravidly [1]

Software nedělá něco, co by podle specifikace produktu dělat měl.

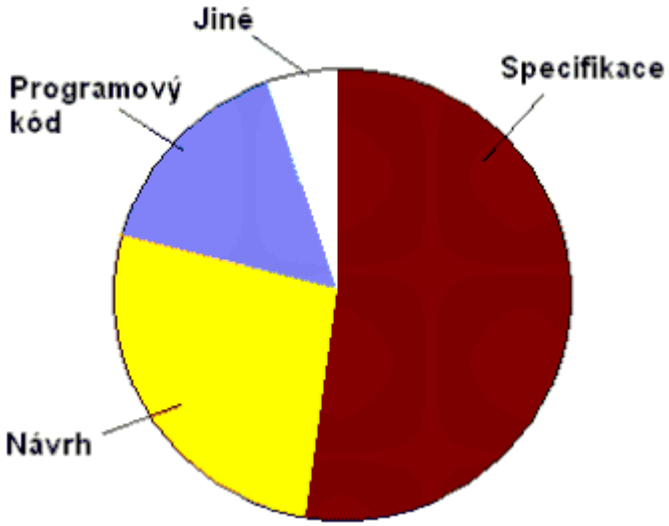
Software dělá něco, co by podle specifikace produktu dělat neměl.

Software dělá něco, o čem se produktová specifikace nezmiňuje.

Software nedělá něco, o čem se produktová specifikace nezmiňuje, ale měla by se zmiňovat.

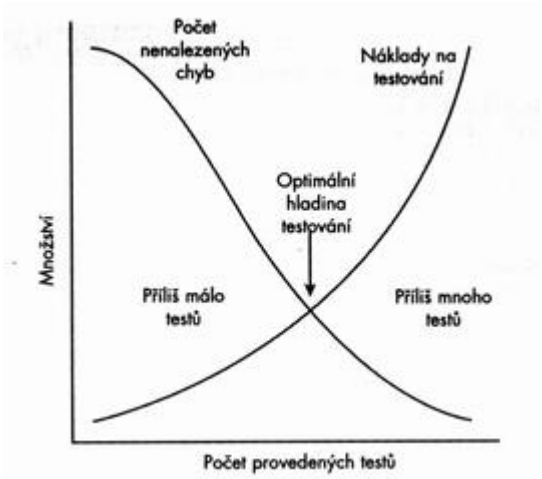
Software je obtížně srozumitelný, těžko se s ním pracuje, je pomalý, nebo - podle názoru testera softwaru – jej koncový uživatel nebude považovat za správný.

Softwarová chyba je velice nepříjemnou věcí a může vzniknout v jakékoli fázi vývoje produktu, jednou z nejčastějších příčin vzniku chyby je specifikace produktu, jak je ukázáno na obrázku 1. To může být způsobeno nedostatečnou přesností požadavků nebo jejich častými změnami ve specifikaci. Další příčinou vzniků chyb je samotný návrh, v něm vznikají chyby podobným způsobem jako ve specifikaci. Návrh může být uspěchaný nebo není dostatečně prokonzultovaný se všemi členy vývojového týmu. Při programování vznikají chyby především díky složitosti vytvářeného zdrojového textu a časovému nátlaku na programátory. [1]



Obrázek 1: Znázornění nejčastějších příčin chyb [1]

Při vývoji produktu chyby vznikají a je potřeba je odstranit, s tím jsou spojené také náklady na odstranění takové chyby. Cena odstranění chyby se liší podle toho, v jaké fázi vývoje se produkt nachází. Čím dříve je chyba nalezena, tím jsou náklady na její odstranění nižší a s přibývajícím časem nalezení chyby cena odstranění znatelně stoupá. Ve fázích specifikace a návrhu produktu je po zjištění chyby její odstranění nejjednodušší a nejlevnější, chyby při vytváření zdrojového textu nebo testování jsou dražší a obtížněji odstranitelné, ale nejhorší možnou situací je, když chybu najde koncový uživatel. Podle Pattona má testování svoji optimální hladinu v poměru nákladů a chyb, který je znázorněn v obrázku 2. [1]



Obrázek 2: Optimání hladina testování [1]

Tato hladina nemusí být vždy objektivním měřítkem pro rozhodnutí, kdy s testováním skončit. Dva různí testeři napíší různé množství testů s různými náklady, a proto také záleží na zručnosti a testera. [4]

2.2. Modely procesu vývoje software

Vývoj software je souhrn aktivit vedoucích k vytvoření výsledného softwarového produktu. Existuje několik modelů, které nám říkají, jak tyto aktivity provádět za sebou abychom úspěšně vytvořili softwarový produkt. [6] Většinou se takový model skládá z kroků jako:

- plánování
- design systému
- implementace, kódování
- testování
- validace.

Uvedu zde tři základní modely vývoje softwaru: spirálový model, přírůstkový model a vodopádový model vývoje. [7] Existují i metodiky agilní založené na přírůstkovém a iterativním vývoji, jedná se o soubor metod pro vývoj softwaru.

2.2.1. Spirálový model

Tento přístup pro vytváření software byl poprvé zveřejněn softwarovým inženýrem Barrym W. Boehmem v roce 1988. [8] Metoda zobrazuje vývoj softwaru jako spirálu a každý jeden cyklus (iterace) této spirály, ukázané na obrázku 3, se skládá se ze čtyř kroků: [6]

Obrázek 3: Znázornění spirálového modelu [9]

- Analýza – stanovení cílů, vytvoření podrobného plány iterace
- Hodnocení – identifikace rizik a jejich řešení, zjištění alternativ
- Vývoj – vývoj části projektu a testování
- Plánování – plánování další iterace

Tento přístup se snaží minimalizovat rizika při vývoji softwaru. [10]

2.2.2. Přírůstkový model

Tento přístup k vývoji byl poprvé navrhnut profesorem Harlanem Millsem roku 1980. [11] Při takovémto přístupu k vývoji se produkt „skládá“ tak, že část projektu se vytváří iterativně, podobně jako u spirálového modelu, a část sekvenčně. Ve výsledku to znamená, že nejdříve se vytvoří základní část vytvářeného softwaru s nejzákladnější funkčností a další části se vytvářejí postupně poté a nabalují se na základ. Výhodou

takového přístupu je možnost poskytnou zákazníkovi základní část softwaru, kterou zákazník může začít rovnou využívat, dále poskytuje flexibilitu při vývoji dalších komponent, zákazník může požádat o změnu nebo o to v jakém pořadí chce další komponenty přidávat. [12]

2.2.3. Vodopádový model

Ve vodopádovém modelu se přistupuje k jednotlivým činnostem sekvenčně, což znamená, že jdou po sobě. Nejdříve se zjistí požadavky zákazníka a vytvoří se specifikace produktu, následuje návrh produktu a poté implementace s prvním testováním. Jednou z posledních fází vývoje je integrace jednotlivých částí systému do sebe, v této fázi se provádí testování. Poslední fází vývoje je provoz a údržba. [10] Vodopádový model vývoje je velice rizikový, protože zákazník vidí produkt až na konci vývoje, přitom se mohly jeho potřeby a požadavky na výsledný software změnit. [6] Proto model není používaný a není vhodný pro vývoj. Tento model zde uvádím pouze z historického hlediska vývoje softwaru.

2.2.4. Agilní metodiky

U agilních metodik vývoje softwaru je kladen důraz na spolupráci mezi vývojáři, dále na funkčnost vyvíjeného softwaru, rychlou reakci na změny v požadavcích od zákazníků a na spolupráci mezi vývojovým týmem a zákazníky. Pro agilní metody byl v roce 2001 vytvořen manifest s prioritami a principy agilního vývoje softwaru. Mezi agilní metodiky patří například extrémní programování a scrum. [13]

2.2.4.1. Extrémní programování

Extrémní programování nejspíše jedna z nejznámějších agilních programovacích metod, přináší časté dodávky softwaru ve velice krátkých vývojových cyklech. Při extrémním programování se vytvářejí jednotkové testy již před napsáním samotného zdrojového textu, aby byl zdrojový text co nejjednodušší a nejpřehlednější. Klade se důraz na komunikaci, zpětnou vazbu mezi všemi zainteresovanými, jednoduchost v zdrojových textech a odvahu vývojáře pustit se do velkých změn. [14]

2.2.4.2. Scrum

Metoda Scrum je založena na iterativním přístupu k vývoji a pracuje v takzvaných sprintech což je časové období, většinou týden, ve kterém probíhá část vývoje produktu, každý sprint začíná plánovací schůzkou a končí závěrečnou schůzí, mezi nimi probíhají každý den Scrum schůzky kde každý z vývojářů zhodnotí předešlý den a uvede svůj rozvrh prací na den stávající. V týmu jsou definovány tři role: [15]

- Product owner – Vymezuje co je zapotřebí ve sprintu udělat.
- Development team – Vývojový tým, který vytvoří a předvede co Product owner zadal.
- Scrum master – Zajišťuje co nejhladší průběh sprintu, snaží se vylepšit proces vývoje produktu.

2.3. Metody testování

2.3.1. Statické a dynamické testování

Rozdíl mezi statickým a dynamickým testováním je v nutnosti testovanou aplikaci spustit. Při statickém testování není třeba běh testované aplikace, hledají se potencionální problémy ve zdrojovém textu nebo vztahy mezi zdrojovým textem a modelem vyvíjeného softwaru. Tato forma testování už v dnešní době není využívána, tak často, jak by měla, protože statické testování je časově náročné. [4] Statické

testování provádí také překladače a sestavovací programy, které kontrolují syntaxi zdrojového textu.

U dynamického testování je zapotřebí běh testovaného softwaru, kterému se předkládají různé vstupy a kontrolují se výstupy, nebo se testuje, jak software zareaguje při neposkytnutí vstupů a jestli výstupy existují. [7]

2.3.2. Černá a bílá skříňka

Metody testování černé a bílé skříňky se od sebe liší ve znalosti vnitřní funkčnosti, proto také můžeme říkat funkční testování, [6] testovaného softwaru. Při testování černé skříňky (Black-Box testing) tester neví a ani ho nezajímá, jak program funguje uvnitř, tester předkládá programu vstupy a kontroluje, zda jsou výstupy správné.

Naopak při testování bílé skříňky (White-Box testing) tester zná strukturu a vnitřní fungování programu, může tak vytvořit testy, které [12]

- zaručeně projdou každou cestu programu alespoň jednou
- u logických výrazů vyzkouší jak pravdivou stranu, tak stranu nepravdivou
- u cyklů zkontrolují hraniční hodnoty
- zkontrolují vnitřní datové struktury a zajistí jejich správnost

2.3.3. Automatické a manuální testování

Při manuálním testování provádí testy člověk, zatímco při automatickém je to počítač, kdo se stará o spouštění a vyhodnocování testů. Manuální testování je vhodné u testů, kde je zapotřebí lidský úsudek nebo časté změny v přístupu k testování, také nemusí-li se testy spouštět opakovaně s velkým množstvím vstupních dat. [4]

Automatické testování je výhodné použít, potřebujeme-li testy spouštět často, vícekrát a s velkým množstvím vstupních dat. Automatickému testování a jeho vhodnosti použití se budu podrobněji věnovat v některé z další části práce.

2.3.4. Stupně testování

Testování můžeme rozdělit do několika základních stupňů podle toho, kdy a v jaké části projektu se testy přidávají do procesu vytváření softwaru. [2]

Stupně testování jsou:

- Jednotkové testování
- Integrační testování
- Systémové testování
- Akceptační testování

Při testování jednotek (Unit testing) se snažíme otestovat pouze část testovaného zdrojového textu. Takovéto testy se vytváří pomocí kódu, takže je možné, aby je vytvářeli rovnou programátoři při vývoji. Testy jsou automatické a dobře se s nimi pracuje. [4]

Integrační testování (Integration testing) zjišťuje, jestli vytvořené komponenty a subsystémy správně spolupracují a jsou správně zapojené. Tyto testy už většinou provádí testovací tým. Testy mohou být manuální i automatizované, záleží na situaci a projektu. [12] [4]

Pro zjištění, jestli celý systém správně funguje, slouží systémové testy (System testing), které se snaží otestovat celý systém a jeho chování. Testy se například mohou chovat jako uživatel nebo se mohou chovat náhodně. Důležité je, aby testy prošly celý systém takzvaně skrz naskrz (back-to-back) a otestovaly všechny možné případy užití programu. [4]

Jedním z posledních stupňů testování je testování akceptační (Acceptance testing), které může být součástí převzetí vytvořeného produktu zákazníkem. Akceptační testy mohou probíhat pomocí scénářů, které jsou odvozené od případů užití, doplněny o požadavky a odkazy, které jsou jimi testovány. Takovéto scénáře jsou zákazníkem odsouhlaseny. [4]

3. Automatické testování

Při tvorbě softwaru se vývojáři většinou řídí některým modelem vývoje, a součástí těchto modelů je i testování, které se může i vícekrát opakovat. A to je jeden z důvodů, proč začít používat automatické testy a automatizované nástroje pro testování. Dalšími důvody může být úspora času a zrychlení testování. Jedná se vlastně o stejné druhy a typy testů jako při manuálním testování, jen jsou testy automaticky vyhodnocovány, nebo automaticky generovány. Jedněmi z nejdůležitějších vlastností testovacích nástrojů a automatizace jsou: [1]

Rychlost – při automatickém testování máme možnost otestovat produkt o mnoho rychleji než při manuálním testování.

Efektivita – při použití automatického nástroje uspoříte čas, který můžete věnovat vymýšlení nových testových případů, k přípravě a plánování dalšího testování. Některé nástroje nabízí i možnost nechat probíhat testy na vzdáleném počítači a tím šetří čas při testování celých balíků testů nebo testů, které potřebují větší dávku času na zpracování.

Správnost a přesnost – testovací nástroje provádí testy přesně tak, jak jsou testy nástroji předloženy, a testy jsou pokaždé přesně a správně provedeny.

Neúnavnost – automatické nástroje se nemohou unavit na rozdíl od testera, jsou schopné samostatně opakovat testy a nezatěžují testera opakováním testů, tester dostane pouze výsledky testů.

Všechny automatické nástroje jsou pouze nástroje, vždy je zapotřebí testera. Tyto nástroje usnadňují práci testera a mohou mít vliv na kvalitu testování, ale testera nenahradí. Specifikace softwaru se stále mění, a tak je potřeba vytvářet automatizované testy flexibilně, aby mohli být použity i při změnách v softwaru. Při použití nástrojů pro tvorbu automatických testů se na automatizaci nikdy moc nespoléhat, protože software nejde otestovat úplně ani za použití automatizace. [1]

4. Přehled nástrojů

Nejprve se seznámíme s několika nástroji pro automatické testování. Každý nástroj má řadu vlastností a funkcí. Nástroje často neslouží pouze k vytváření testů nebo testovacích skriptů, ale podporují i další funkce, jako je správa projektu, workflow, nebo management testování. Může se jednat o jednoduchý program pro vytváření testů nebo také o celé řešení pro střední a velké podniky. Cílem této práce je najít nástroje, které automaticky vytváří testy nebo alespoň jejich část, a tím usnadňují práci člověku, který zajišťuje testování.

4.1. Hledané charakteristiky

V popisu nástrojů se zaměřím na několik charakteristik těchto nástrojů. Základní charakteristiky:

- Licence
- Zdroj
- Podporovaná technologie
- Druhy testů
- Míra automatizace
- Stáří nástroje
- Podpora nástroje

Zajímavou charakteristikou je míra automatizace, často je rozdílná, některé nástroje za automatizaci považují automatické spouštění skriptovacích testů, které napsal tester, jiné naopak automatizují práce spojené s testováním jako kontrolu, administrativu a některé mohou generovat přímo testovací skripty.

4.2. Vybrané nástroje

V tabulce 1 je vypsáno několik nástrojů pro automatické testování včetně dostupnosti nástroje v trial nebo demo verzi a s případnou cenou licence. Některé společnosti ceny svých licencí veřejně neuvádějí, místo toho ceny vytvářejí na míru zákazníkům a neposkytují informace o cenách svých produktů. Dále jsou uvedeny ke každému nástroji některé charakteristiky zvlášť a krátký popis jaké všechny druhy testů nástroj podporuje a jaké služby poskytuje.

Nástroj	Dostupnost	Případná cena licence (Všechny ceny jsou za jeden kus licence)
Conforminq Designer	Bez trial nebo demo verze	Cena na míru
Test Studio	S trial verzí	2 999$ 199 $ měšíčně
Peta	S demo verzí	Designer: 1 960 € Core: 5 950 € Load: 7 950 € Webrecorder: 990 €
Jubula	Dostupný po registraci	Zdarma
AgitarOne	S trial verzí	Cena na míru
Randoop	Dostupný	Zdarma
Concordion	Dostupný	Zdarma

Tabulka 1: Přehled všech prozkoumaných nástrojů s dostupností a případnou cenou

4.2.1. Conformiq Designer

Charakteristika	Popis
Technologie	C#. C++, Java, Perl, Python, TTCN3, Juint, HTML, Visual Basic, Shell Selenium, HP Quick Test Profesional
Testy	Black-box a white box testy, unit testy, integrační, systémové
Automatizace	Automatické generování testovacích skriptů
Licence	Komerční
Verze a stáří	v4.4.3 (2012)
Zdroj	http://www.conformiq.com/

Tabulka 2: Charakteristiky nástroje Conformiq Designer

Conformiq Designer je nástroj pro vytváření testovacích skriptů pomocí modelů testovaných systémů. Tester vytvoří model předpokládaného chování systému a nástroj tento model převede v sadu testovacích skriptů i se vstupními a výstupními daty, které se pak dají aplikovat na vyvíjený software. Modely se vytvářejí v jazyce UML, k tomu společnost Conformiq nabízí nástroj Modeler [16] zcela zdrama, ale podporuje i jiné nástroje k tomu určené jako například IBM Rational nebo může být model pouze textový vytvořený použitím Conformiq Modelling Language (QML) [17].QML je jazyk odvozený od jazyka Java vytvořený společností Conformiq. Nástroj sám testy pouze generuje a dokáže testy exportovat do jednoho z podporovaných formátů nebo rovnou vytvořit spustitelné balíčky testů. Nástroj je dobrý ke generování funkčních, integračních a systémových testů. Cena nástroje Conformiq Designer pro komerční užití se stanoví až po jednání se společností Conformiq. Je možnost bezplatné licence na nástroje společnosti Conformiq pro akademické uživatele nebo pro ty, kteří jsou ochotni

zveřejnit své modely. Nástroj je stále společností podporován, ale společnost se nejvíce zaměřuje na podporu svých zákazníků a veřejnosti nevěnuje moc pozornosti. [18]

4.2.2.Test Studio

Charakteristika	Popis
Technologie	HTML 5, XAML, Ruby, AJAX, Java, PHP
Testy	Funkční, zátěžové, výkonové
Automatizace	Nahrávání a automatické spouštění testů
Licence	Komerční
Verze a stáří	V2014.1.410.0 (2014)
Zdroj	http://www.telerik.com/automated-testing-tools/

Tabulka 3: Charakteristiky nástroje Test Studio

Test Studio je nástroj pro testování webových a mobilních aplikací, jak aplikací pro iOS, Windows Phone nebo Android, tak webových aplikací, které jsou určené pro zobrazení na mobilním zařízení. [19] Nástroj poskytuje jednoduché nahrávání testů přímo v prohlížeči, ale i manuální psaní skriptů a jejich následnou přeměnu v testy automatické. Umí pracovat s několika prohlížeči, a tak není nutné, upravovat testy pro každý prohlížeč zvlášť. Celá řada funkcí umožňuje nahrávat testy, spouštět je a schraňovat na jednom místě. Poskytuje možnost funkčního testování, zátěžových testů a výkonových testů pro webové aplikace. Je možné automaticky přetvářet funkční testy na testy zátěžové, a tím zrychlit průběh testování. Nástroj je celým řešením pro testování při vývoji webových aplikací. Společnost uvedla nástroj jako produkt na trh, a tak stanovila i neměnné ceny. Test Studio je stále společností podporovaný nástroj, na který stále vycházejí aktualizace. Na fóru k nástroji je mnoho dotazů od uživatelů, ale i odpovědí od samotných vývojářů. [20] [21]

4.2.3. Peta

Charakteristika	Popis
Technologie	Java, HTTP
Testy	Systémové, integrační, regresní, zátěžové, stress testy
Automatizace	Automatické generování skriptů, spouštění, nahrávání testů
Licence	Komerční
Verze a stáří	v2.4.0 (2010)
Zdroj	http://www.peta-testautomation.com/

Tabulka 4: Charakteristiky nástrojů Peta

Společnost PETA vytvořila několik nástrojů, které pomáhají s testováním software – Designer, Webrecorder, Core a Load – hlavně s testováním client-server a webových aplikací. Každý z těchto nástrojů má jiné použití, ale jsou propojeny přes nástroj Core, který slouží jako jádro celého systému, sdružuje všechny testy a data vytvořená v rámci ostatních nástrojů. Všechny nástroje jsou vytvořeny v jazyce Java, a proto jsou spustitelné na všech platformách. Společnost PETA stále pracuje na svých nástrojích a ty jsou stále podporovány. Ceny jsou nastaveny u některých nástrojů podle počtu licencí, každá za jeden počítač nebo uživatele, a u jiných pevně za použití nástroje ve firmě.

Designer – nástroj sloužící k modelování testů na základě výměny zpráv mezi oběma aplikacemi. Testy se modelují přidáváním jednotlivých událostí, operací a reakcí na ně buď na straně klienta, nebo na straně serveru. Je možné namodelovat několik různých testů a všechny je najednou spustit. Výsledkem jsou XML skripty, které se dají ještě manuálně upravit, pozměnit a samozřejmě také dále použít.

Webrecorder – další z nástrojů je na nahrávání testových modelů přímo v prohlížeči. Ve Webrecorderu nastavíte nahrávání a pak v prohlížeči provedete úkony, které chcete nahrát, výsledkem je skript se všemi úkony, které byly v prohlížeči provedeny, dále je možno s tímto skriptem dále pracovat.

Core - je srdcem celé testovací platformy, tento systém je zodpovědný za spouštění všech testovacích XML skriptů vytvořených ostatními nástroji z rodiny PETA.

Load – pomáhá při spouštění a vytváření zátěžových a stress testů. Lze simulovat zátěž serveru, zjišťovat kolik maximálních požadavků server zvládne obsloužit. Load dokáže také simulovat realistické chování uživatele na serveru. [22]

4.2.4. Jubula

Charakteristika	Popis
Technologie	Java, HTML, iOS
Testy	Black-Box testy
Automatizace	Vytváření testů, spouštění a analýza výsledků
Licence	EULA
Verze a stáří	V8.0. (2014)
Zdroj	http://testing.bredex.de/

Tabulka 5: Charakteristiky nástroje Jubula

Jubula je nástroj pro automatické testování grafického rozhraní (GUI), pomáhá při funkčním GUI black-box testování. Jubula vytváří testy pro Java, HTML a PHP aplikace pomocí drag&drop. Jednoduše vyberete akci, jako například kliknutí na tlačítko, zadání vstupní hodnoty a podobně, kterou má skript provést a zařadíte do seznamu akcí, po dokončení je možné test ihned spustit. Při vytváření testu si musíte vytvořit i vlastní vstupní a výstupní data. Testy se nedají vygenerovat do externího skriptu a použít ho samostatně nebo v jiném testovacím prostředí. Jubulu je možné stáhnout na domovských stránkách zdarma po registraci, jako samostatnou aplikaci, nebo nainstalovat plug-in do vývojového prostředí Eclipse. Jubula je společností stále podporována, tomu nasvědčuje i aktualizace vydaná 16.4.2014 [23], většinu otázek nebo problémů lze dohledat na fóru vztahující se k nástroji Jubula. [24]

4.2.5. AgitarOne

Charakteristika	Popis
Technologie	Java
Testy	Unit testy, regresní
Automatizace	Automatické generování unit testů
Licence	Komerční
Verze a stáří	v5.5 (2013)
Zdroj	http://www.agitar.com/solutions/products/agitarone.html

Tabulka 6: Charakteristiky nástroje Agitar One

Agitar One je nástroj pro automatické generování a spravování jednotkových testů dokáže vytvořit celou sadu jednotkových testů, přímo ze zdrojových textů, pro testovaný program napsaný v jazyce Java. Nástroj dokáže automaticky vytvořit jednotkové testy, které odrážejí stávající stav zdrojových textů na základě analýzy zdrojového textu, také automaticky generuje vlastní vstupní data, pro testy a analyzuje výsledky testů. Nástroj vstupní data generuje podle vygenerovaných testů a je možné nastavit rozsahy generovaných dat testerem. Jednoduché generování jednotkových testů usnadňuje práci testerovi, nástroj přehledně zobrazí, ke kterým částem kódu vygeneroval jednotkové testy a které části jsou zatím testy nepokryté. Společnost poskytuje ceník svých služeb pouze svým klientům a nikomu jinému. Nástroj je stále podporovaný ze strany vývojářů ale nenalezl jsem žádné fórum nebo diskuse o nástroji, pouze několik článků na osobních blozích. [25] [26]

4.2.6. Randoop

Charakteristika	Popis
Technologie	Java
Testy	Unit, regresní
Automatizace	Automatické generování unit testů
Licence	MIT License
Verze a stáří	v1.3.4 (2014)
Zdroj	https://code.google.com/p/randoop/

Tabulka 7: Charakteristiky nástroje Randoop

Randoop je nástroj pro automatické generování jednotkových testů ve formátu JUnit. Nástroj pouze testy generuje, a proto je zapotřebí ještě knihovna JUnit pro spouštění a zpracování testů. Randoop je velice rychlý pomocník při testování software a dokáže usnadnit práci testerovi tím, že vytvoří alespoň základní jednoduché JUnit testy. Používá metodu zpětného řízení generování testů, tato metoda náhodně generuje sekvence metod a volání pro třídy, které mají být otestovány. Tyto sekvence použije pro vytvoření testů, testy spustí a na základě výsledku vytvoří tvrzení, kterými zachytí chování programu a zachytí chyby. Nástroj lze stáhnout zdarma ze stránek projektu buď jako spustitelnou distribuci, která stačí ke generování testů, nebo je možné nainstalovat plug-in do vývojového prostředí Eclipse. Randoop je stále vyvíjen. [27]

4.2.7. Concordion

Charakteristika	Popis
Technologie	Java
Testy	Akceptační
Automatizace	Vytváření akceptačních testů
Licence	Apache Licese v2.0.
Verze a stáří	v1.4.4. (2013)
Zdroj	http://www.concordion.org/

Tabulka 8: Charakteristiky nástroje Concordion

Nástroj Concordion používá upravené specifikace vyvíjeného software jako akceptační testy. Specifikace se napíše přirozeným jazykem za použití HTML značek. Specifikaci poté tester upraví přidáním speciálních HTML značek k popisu funkčností, které chce otestovat, a vytvoří spojku v jazyce Java, mezi testem a systémem k otestování. Spojka se chová jako buffer mezi specifikací a testovaným systémem, také je JUnit testem tohoto systému. Při spuštění spojky jako JUnit test, vytvoří nástroj jako svůj výstup HTML soubor, který bez dalších úprav zobrazí výstup JUnit testu do specifikace produktu. Rozdíly mezi skutečným a očekávaným výstupem testu zvýrazní červeně. Testovaný software je stále provázán se specifikací, a tak je hned vidět, jestli je implementace nebo změna v implementaci ve shodě se specifikacemi softwaru. Výhodou tohoto systému testování je, že testy se tvoří už při specifikování chování systému. Značkovací jazyk HTML umožňuje psát specifikace i lidem bez velkých znalostí programování. Nástroj Concordion je volně ke stažení na domovských stránkách projektu a k nástroji jsou dále vytvářena rozšíření. [28]

5. Podrobné seznámení s vybranými nástroji

V následující kapitole se seznámím s několika vybranými nástroji podrobně. Pro podrobný popis jsem vybral nástroje Test Studio, Jubula a Randoop, kde každý z nástrojů vytváří testy jinak, a také jejich různé druhy. V případě Test Studia se jedná o robustní nástroj pro testování webových aplikací a webových stránek, který dokáže zastat mnoho funkcí a poskytuje celou řadu možností, jak webové aplikace a webové stránky testovat. Jubula se zaměřuje na grafické rozhraní aplikací, které je v dnešní době již nedílnou součástí téměř každé aplikace. Posledním nástrojem je Randoop, který pomáhá při jednotkovém testování a svou jednoduchostí je velice užitečným a rychlým pomocníkem.

5.1. Popis nástrojů

U všech nástrojů se zaměřím na kritéria vypsaná v tabulce 9 a stručně popíši práci s každým nástrojem.

Kritérium	Popis kritéria
Práce s nástrojem	Náročnost práce s nástrojem, základní úkony
Vytváření testů	Způsob vytváření testů
Editace testů	Jak je možné testy upravovat a přizpůsobovat
Spuštění a opakování	Možnosti spouštění testů a jejich opakování
Možnosti testování	Jaké aplikace a jak je možné testovat
Cena	Cena nástroje a licence

Tabulka 9: Kritéria nástrojů

5.1.1. Test studio

Nástroj Test Studio vytvořila společnost Teletrik, která se zabývá vytvářením nástrojů pro zvýšení produktivity v týmu, komponent grafického rozhraní, také nástrojů pro automatizaci testování webových aplikací a stránek. Nástroj Test Studio je produktem, který se specializuje na automatizaci testování. Společnost si v roce 2002 založilo několik přátel, v současné době je to společnost s rozsáhlým portfoliem a pokrytím trhu, jejím mottem je „Poskytnout více než je očekáváno.“ [29]

5.1.1.1. Práce s Test Studiem

Práce s Test studiem je jednoduchá a příjemná, všechny ovládací prvky jsou čitelné a dobře rozložené. Jako první vás uvítá nabídka, kde si vyberete, na jakém projektu chcete pracovat, projekt můžeme označit za soubor testů vztahujících se

k stejné aplikaci či webové stránce. Projekty se ukládají na pevný disk do složek, kam si při instalaci nastavíte. Po vybrání projektu přejdete na stránku projektu, kde jsou všechny testy pro daný projekt na jednom místě. Po několika minutách používání nástroje není problémem cokoli nastavit, vyhledat nebo spustit. Vývojáři nejspíše kladli veliký důraz na jednoduchost a možnost rychlého osvojení používání nástroje. Pro začínajícího uživatele je velice vhodná záložka „Help“, kde je na jenom místě mnoho materiálu od uživatelských fór až po videa s tutoriály.

5.1.1.2. Vytváření testů v Test Studio

Vytváření testů je možné dvěma způsoby, pomocí automatického nahrávání nebo skládat manuální kroky testu za sebe. Automatické nahrávání testů je nepochybně zajímavější možností vytváření testů. Webové testy se dají vytvářet pomocí nahrávání kroků testování přímo v prohlížeči, samotný nástroj podporuje prohlížeče Internet Explorer, Chrome, Firefox a Safari. Po kliknutí na ikonu „Record“ se spustí prohlížeč s několika základními informacemi pro nahrávání testů. URL adresa testované stránky se zadává do spodní lišty, kde se nachází i tlačítko pro pozastavení testu, tlačítko kontextové nabídky a tlačítko pro zapnutí označování elementů na stránce. Po nahrání testu jsou všechny provedené kroky vidět v seznamu za sebou, jak byly provedeny.

5.1.1.3. Editace tesů v Test Studio

Po vytvoření testu je možno test ještě editovat a přidávat do nich různé další kroky testu, logické výrazy nebo vypořádání se s dialogy, které se mohou v testu objevit. U samotných nahraných kroků lze také upravit mnoho nastavení, všechna jsou přehledně zobrazena v pravé části obrazovky, ty nejdůležitější po rozkliknutí daného kroku. Do testů je možné přidávat i vlastnoručně naprogramované kroky, dají se napsat v C# nebo Visual Basicu. [30]

5.1.1.4. Spouštění a opakování testů v Test Studio

Po vytvoření testu je možné test zařadit do seznamu testů, kde se shromažďují testy připravené k naplánování testování. Plánování testování je jednoduché, stačí jen vybrat seznam testů, které se mají provést, naplánovat jim přesný čas a datum provedení, také lze nastavit opakování testů. Je na výběr mnoho možností plánování od jednoho testování v určitý den až po opakování testů v určitý den a hodinu. Další možností je vybrání testovacích strojů, na kterých se budou testy provádět.

5.1.1.5. Možnosti testování v Test Studio

Test Studio nabízí široký výběr technologií, které může nástroj testovat výčtem HTML 5, XAML, Ruby, AJAX, Java a PHP. [21] Testy pracují přímo s prohlížečem a na implementaci technologií nekoukají, ale dokážou vyvolat situace, kdy je technologie vyžadována. Umožňuje testovat webové aplikace v Internet Exploreru, Chrome, Firefox a Safari. V Test Studiu můžeme provádět funkční testy, testy výkonu a zátěžové testy. Jedná se o black-box testování s možností importovat vlastní testovací data z Excel souborů. U zátěžových testů nabízí Test Studio virtuální uživatele a umožňuje nastavit jejich množství i to, jak se bude jejich počet v čase zvyšovat, a rozdělení zátěže mezi jednotlivé skupiny uživatelů. U výkonových testů můžeme určovat, jaké veličiny

chceme sledovat. Například aktivitu fyzické síťové karty, odeslané a přijaté diagramy protokolů IPv4 a IPv6, počet operací čtení souboru, aktivitu procesoru sledovaného zařízení a mnoho dalších veličin. Test studio má i svůj doplněk do vývojového prostředí Visual Studio, s kterým spolupracuje a umožňuje rovnou přenést reporty chyb k vývojářům a testerům. [21]

5.1.1.6. Cena Test Studia

Test studio je komerční produkt určený na trh, a tak je možné si ho stáhnout a zakoupit licenci na stránkách společnosti Telerik, kde je také ke stažení třiceti denní trial verze. Po vyčerpání trial verze produktu můžete zakoupit buď neomezenou licenci, nebo platit předplatné. Předplatit licenci je možné na celý rok, s nemožností zrušení předplatného, nebo platit za každý měsíc užívání zvlášť. [21]

5.1.2. Jubula

Jubula vzikla jako doplněk do vývojového prostředí Eclipse, poté se firma BREEDEX GmbH k projektu připojila a přetvořila Jubulu do samostatné aplikace, současně vytvořila i téměř stejný nástroj Gui Dancer. Společnost se zaměřuje na vytváření konečných řešení pro své zákazníky, konzultace, testování a školení. Také spolupracuje s open-source komunitami a pomáhá jim s vývojem jejich softwaru. [23]

5.1.2.1. Práce s nástrojem Jubula

Při prvním spuštění nástroje velice potěší podobnost s vývojovým prostředím Eclipse, takže pro ty, co mají s Eclipse zkušenosti, není ovládání Jubuly ničím složitým. Samotné rozvržení pracovní plochy může být z počátku nepřehledné, ale po vyzkoušení a vytvoření jednoho či dvou projektů lze nástroj využívat bez větších problémů. Jedinou nepříjemností při používání Jubuly může být nastavení Application Under Test agenta (AUT agent), které není úplně jednoduché a zasloužilo by si nápovědu nebo tipy pro použití přímo v dialogu pro nastavení AUT agenta. Jubula disponuje nápovědou a jednoduchými návody na vytvoření prvního projektu, které jistě ocení každý, kdo s Jubulou začíná pracovat. Jednou z nevýhod používání Jubuly je možnost exportovat vytvořené projekty pouze do XML souborů a ne do některého z podporovaných programovacích jazyků.

5.1.2.2. Vytváření testů

Při vytváření testů v nástroji Jubula je důležité samotné spojení testované aplikace s nástrojem, pro to slouží AUT agent, který testovanou aplikaci spustí a zajistí funkčnost služeb nástroje Jubula. Nastavení agenta může být v některých případech poněkud složitější, ale z mé zkušenosti je možné připojit téměř jakoukoliv podporovanou aplikaci, lze také připojit i více aplikací k jednomu projektu. Samotné vytváření testů je založeno na principu drag&drop, nástroj dává na výběr mnoho různorodých akcí, jako například kliknutí myší, vstup z klávesnice a ověření viditelnosti grafické komponenty. U některých akcí nelze nastavit přesnou pozici v okně, a proto nelze ovládat úplně vše, například k testování posuvníků jsem v nástroji žádnou akci nenalezl. Po vytvoření seznamu akcí je potřeba akce pojmenovat, pojmenovat jejich parametry a přidat vstupní data buď manuálně, nebo z centrální databáze. Takto vytvořené testy lze sdružovat do testovacích balíčků, a tak je i spouštět. Posledním úkonem je mapování GUI testované aplikace a vytvořením popisu grafických komponent použitých v testech. I v tomto ohledu je Jubula velice jednoduchá, po zapnutí nahrávání stačí jen ukázat kurzorem na požadovaný prvek a zelené ohraničení ukáže, zda je prvek mapovatelný. K zaznamenání prvku GUI slouží klávesová zkratka Control+Shift+Q. Po namapování prvků je test připraven k spuštění.

5.1.2.3. Editace testů

Vytvořené testy lze měnit jednoduchým smazáním akce a přidáním nové, problémem jsou již nastavené parametry pro test a jejich vstupní data. Ta po smazání akce nezmizí sama a zůstanou v testu, což může působit problémy. Přehlednější, zato pomalejší a zdlouhavější cestou je smazat celý test a vytvořit jej znovu s požadovanou novou akcí. V tomto ohledu je práce s Jubulou nepraktická a může způsobit nepříjemné problémy i při malých změnách v testech.

5.1.2.4. Spouštění a opakování

Jak jsem již uvedl v části o vytváření testů, testy se spouští přes AUT agenta a spuštěný agent je podmínkou pro provedení testů. Další podmínkou je spuštěná aplikace přes agenta. Po spuštění testů nástroj využívá funkce operačního systému, a tak není možné s počítačem ve chvíli průběhu testů pracovat. Testy je možné neomezeně opakovat, ale ne automaticky, každé opakování musí spustit tester. Nástroj sice obsahuje záložku plánování, ale to spíše ve smyslu task managementu a reportování nalezených chyb. Nevýhodou Jubuly je nemožnost nastavit spouštění testů automaticky v určitou dobu. Ale Jubula spolupracuje s aplikací Jenkins, která takovéto plánování testů umí. [31], [32]

5.1.2.5. Možnosti testování

Jubula poskytuje možnost testovat GUI aplikací vytvořené v jazyce Java za pomocí Standard Widget Toolkit (SWT), Swing, RCP a také GUI vytvořené pro iOS HTML a Microsoft Windows. Testuje funkčnost vytvořeného GUI metodou Black-Box. Nástroj má možnost importovat testovací data z Excel souborů do své centrální databáze, kde je schraňuje a zpřístupňuje pro projekty. [23]

5.1.2.6. Cena

Nástroj Jubula je zaštítěna licencí EULA a je k stažení po registraci na stránkách vývojáře. Jubula má i svůj placený protějšek s názvem Gui Dancer, podle specifikací vývojářů jsou obě aplikace svou funkčností stejné, rozdíl je v asistenční službě a podpoře školení od vývojového týmu. Jubula je svou cenou i funkčností vhodným nástrojem pro malé vývojářské týmy nebo pro komunitní či studentské projekty a to je, podle mého názoru, místo Jubuly ve světě nástrojů pro testování GUI. [23]

5.1.3. Randoop

Vznikl jako projekt na MIT, s cílem vytvořit nástroj pro generování solidních JUnit testů bez většího úsilí testera. Projekt se ubírá dobrým směrem a už je velmi dobře použitelný pro základní podobu jazyka Java. V této chvíli využívají nástroj Randoop i společnosti jako ABB a Microsoft, tímto má nástroj nastartovanou slušnou budoucnost a využití v oblasti testování. [27]

5.1.3.1. Vytváření testů

Při vytváření jednotkových testů nejen s nástrojem Randoop je potřeba mít k dispozici zdrojový text testované aplikace. Při použití spustitelného balíčku nástroje se testy vytváří zadáním příslušného příkazu a parametrů do konzole nebo příkazové řádky. Nástroj po zpracování zdrojového kódu vygeneruje několik souborů, jeden soubor pro nastavení samotného generování testů a další už s vygenerovanými testy. V případě, že chceme generovat jednotkové testy pro více souborů se zdrojovým

textem, stačí vytvořit textový soubor s názvy tříd, které chceme testovat. Takovýto soubor můžeme využít i v případě, že nechceme generovat testy pro celý zdrojový text, ale jen pro některé třídy či metody obsažené ve zdrojovém textu. Generování testů pomocí plug-inu pro vývojové prostředí Eclipse je z mého pohledu ještě jednodušší než pomocí spustitelného balíčku. Pouze spustíme projekt jako vstupní data pro plug-in Randoop a nástroj se zeptá, pro jaké balíčky, třídy nebo metody chceme vygenerovat testy a kolik testů chceme nechat vygenerovat. Nástroj vytvoří novou složku v projektu, kde jsou všechny vygenerované testy pohromadě a připravené k spuštění pomocí knihovny JUnit.

5.1.3.2. Editace testů

Protože Randoop je nástrojem, který pouze generuje unit testy do externích souborů, můžeme k editaci použít jakýkoli textový editor, nejlépe takový, který dokáže rozeznat syntaxi jazyka Java, abychom měli v kódu přehled. Při použití plug-inu do prostředí Eclipse je možné testy editovat přímo v něm. Vygenerované testy jsou dobře čitelné a dobře strukturované, což usnadňuje jejich editaci.

5.1.3.3. Spouštění a opakování

Sám nástroj testy spouštět neumí, umí testy pouze generovat. Ke spouštění testů bez velkých úprav slouží nástroj JUnit, který testy spustí a podá jejich výsledky. Testy lze samozřejmě spustit několikrát a používat stále znovu. Nástroj Randoop při generování tesů používá pro výstupní soubory stále stejná jména, a tak je možné, že při generování nové sady testů můžete staré testy přepsat novými. Testy po vytvoření mohou být použity jako testy pro vyhledání chyby nebo jako regresní testy.

5.1.3.4. Možnosti testování

Nástroj dokáže vygenerovat testy pro aplikace napsané v jazyce Java. Randoop konkrétně kontroluje několik specifických příčin chyb v aplikacích. Kontroluje, zda objekt nenavrací hodnotu null, kontroluje návratovou hodnotu objektu, symetrii mezi objekty při použití metody equals, rovnost hash kódu při použití metody hashCode a „No null pointer exceptions“výjimky. Nástroj nejdříve zdrojový text spustí a zjistí, jak se program chová, podle jeho chování vytváří pravidla pro testování. [33]

5.1.3.5. Cena

Nástroj je volně k stažení na stránkách projektu nebo jako stažitelný doplněk pro prostředí Eclipse, také zdarma. Randoop je využitelný jak pro osobní, studentské, nebo komunitní projekty ale, i pro softwarové firmy, kde je schopen pomoct při vývoji a testování Java aplikací. [27]

6. Vytvoření ukázkových případů užití

Po podrobnějším seznámení s vybranými nástroji jsem vytvořil pro každý nástroj ukázkový případ použití. K ukázce nástroje bylo zapotřebí vytvořit nebo najít vhodné ukázkové příklady. Pro nástroj Test Studio jsem vytvořil jednoduché webové stránky za použití technologií HTML, CSS, PHP a Java Script. Stránky se chovají jako velice jednoduchý obchod s knihami, už při vytváření stránek jsem nástroj použil, abych otestoval jejich funkčnost. U nástroje Jubula jsem použil aplikaci Java File Manger [34], abych na ní mohl ukázat práci s co nejvíce grafickými komponentami, které nabízí knihovna Swing. Pro nástroj Randoop jsem vytvořil několik metod za použití jazyka Java, některé z metod mají přímo vytvářet chybové hlášení, že se v metodě nachází chyba. Dále jsem použil příklad deadlocku [35] a problému producenta a konzumenta. [36]

Při psaní ukázkových případů jsem se snažil o popsání postupu při vytváření testů pro ukázkové aplikace a při tom ukázat, jak s nástroji pracovat i obecně. Případy jsem psal velice podrobně, aby je mohl použít i někdo, kdo s nástroji nemá zkušenosti.

Pro nástroje Test Studio a Jubula jsem vytvořil i několik ukázkových videí, které přímo zobrazují postup při vytváření ukázkových případů. Takovéto vizuální zobrazení postupu dle mého názoru pomůže lépe pochopit práci s nástrojem.

Pro práci s nástrojem Test Studio jsem použil aplikaci XAMPP [37], která vytváří Apache server s již připravenými technologiemi MySQL, PHP a Pearl. Použití XAMPP bylo důležité při vytváření zátěžových testů a to z důvodu nezatěžování veřejné sítě.

6.1. Ukázkový případ použití nástroje Test Studio

K ukázkovému případu pro nástroj Test Stuido, jsem vytvořil jednoduché HTML stránky simulující obchod s knihami. Pro vytvoření funkčních web testů je možné použít verzi vyvěšenou na *home.zcu.cz/~veselovs/*, v této verzi stránek je vytvořena záměrná chyba ve výpočtu celkové sumy objednávky, chyba je zde pro ukázku jak na ni nástroj zareaguje. K případu je přiložen excelovský dokument se vstupními daty a složka se všemi soubory testů. Složku testů můžete otevřít přímo z nástroje.

6.1.1. Instalace

Instalačního klienta lze stáhnout z domovské stránky nástroje [21], k ukázkovému případu jsem použil trial verzi nástroje. Po stažení, instalačního klienta spustíme a pokračujeme podle instrukcí. Pro aktivaci nástroje je potřeba mít vytvořený uživatelský účet na stránkách vývojáře, registraci je možné provést i z instalačního klienta.

6.1.2. První spuštění

Při prvním spuštění se nástroj zeptá, jakou trial verzi nástroje chceme odemknout, zvolíme volbu Test Studio Ultimate, abychom mohli využívat jak část nástroje pro funkční testování, ale i část pro zátěžové testování.

6.1.3. Vytvoření projektu

V úvodním menu můžeme vytvořit nový projekt, otevřít existující nebo otevřít ukázkový projekt. Při vytváření nového projektu zvolíme jeho název a umístění na pevním disku, v tomto umístění se budou ukládat všechna data spojená s projektem do

složky s názvem projektu. Ve vytvořeném projektu můžeme vytvářet složky, a tím své testy uspořádat, tato hierarchie zůstává zachována i v úložišti na pevném disku.

6.1.4. Vytvoření testu

Nástroj umožňuje vytvářet funkční, zátěžové a testy výkonu. K vytvoření automatizovaného funkčního testu nástroj nabízí možnost testy nahrát přímo v prohlížeči a kroky testu manuálně upravit, buď v jazyce C# nebo Visual Basic, v obou případech je k dispozici rozhraní (API) od vývojářů nástroje. [30]

6.1.5. Vytvoření funkčního testu

Pro vytvoření funkčního webového testu klikneme pravým tlačítkem na název projektu, vybereme „Add New Test“ z menu, které se ukáže, vybereme první možnost „Web“ a test pojmenujeme. V této chvíli již můžeme začít nahrávat kroky testu, pro to slouží ikona v levém horním rohu s popisem „Record“. Po stisknutí této ikony musíme vybrat prohlížeč, ve kterém budeme test nahrávat. Pro další kroky jsem vždy použil Internet Explorer a to z důvodu, že není nutné žádné další nastavení v prohlížeči. Nástroj se připojí k oknu prohlížeče a začne nahrávat všechny akce, které v prohlížeči vykonáme, pro ovládání slouží ovládací lišta v levém horním rohu, umožňuje nahrávání pozastavit, zvýraznit elementy užité na webové stránce a další ovládací prvky. Pro pohodlnější nahrávání testů a určování elementů stránky, doporučuji zapnout zvýrazňování. Pro ukázku jsem vytvořil HTML stránky a použil aplikaci XAMPP [37] pro vytvoření Apache serveru lokálně na počítači z důvodu nezatěžování veřejné sítě. Na stránku přistoupíme zadáním její URL do navigační řádky prohlížeče, už tímto se v testu vytvoří krok pro přístup na testovanou stránku. Dále můžeme na stránkách provádět jakékoli akce, při zapnutém zvýrazňování elementů se při delším podržení myši nad elementem zobrazí modré tlačítko, po kliknutí na tlačítko se otevře kontextové menu dalších akcí, které můžeme při testování použít jako zobrazení hierarchie stránek, využití elementů stránky, ověřování elementů, ověřování obrázků, využití Java Scirpu, akce myší, drag&drop akce a posouvání se na stránce. Po přistoupení na první stranu doporučuji nechat zapnuté nahrávání, vyplnit položky jméno, příjmení a odeslat tlačítkem potvrdit. Na další straně otevřeme kontextové menu nad obrázkem uprostřed strany a zvolíme ověření obrázku. V menu vyplníme procentuální možnou chybu v zobrazení a potvrdíme. Dále vyplníme položky množství a adresa podle vlastního úsudku. Z menu vybereme knihu, zaškrtneme pole pro zaslání poštou a odešleme. Na další straně otevřeme kontextové menu nad vypočtenou cenou objednávky a zvolíme vytvoření ověření, kde v záložce obsah změníme položku „Exact“ na „Contains“ a do očekávaného textu vyplníme pouze vypočtenou cenu objednávky. Obdobným způsobem ověříme viditelnost tlačítka potvrdit, na poslední straně. Dále můžeme ověřit obsah atributů použitých v odkazu na úvodní stranu použitím záložky atributy v menu pro vytvoření ověření elementu strany. Takto lze vytvořit jednoduchý funkční test pomocí režimu nahrávání. Do vytvořených testů lze vložit i logické výrazy if else, smyčka a cyklus while, vše je schováno pod položkou „Logical“ v navigačním menu funkčního webového testu. Po vložení logického výrazu if else do testu musíme vybrat podmínku výrazu, ta se vybere kliknutím na hvězdičku v pravé části řádku s výrazem if . Tím se v testu se označí hvězdičkou všechny kroky, které lze použít jako podmínku. Jeden krok testu označený hvězdičkou vybereme. Nástroj také zvládá práci s dialogy: upozornění, potvrzení, přihlášení k účtu, stahování a nahrávání. Dialog stačí při nahrávání pouze vyplnit či odkliknout a nástroj vše nahraje, nebo vložit krok k zpracování dialogu ze záložky „Dialogs“ v navigačním menu testu.

6.1.6. Data pro testování

K funkčnímu testu lze přidat i testovací data. Data zle vytvořit manuálně pro každý test zvlášť nebo data importovat jako datový zdroj. Test Studio si poradí se soubory ve formátu xls, CSV, XML, a také dokáže data odebírat z databáze. K propojení, přidání a správě dat slouží sekce „Data Source“, v hlavní nabídce každého projektu. Každý projekt má vlastní datové zdroje a nesdílí je s ostatními projekty. V testu se data ke kroku testu přidají v tabulce „Propreties“ položkou „Bindings“, pod kterou jsou všechny datové proměnné z datového zdroje.

6.1.7. Zátěžový test

Pro vytvoření zátěžového testu, zvolíme při vytváření nového testu možnost „Load“. V první řadě je důležité nastavit plánovacího manažera v záložce „Manage“. Jako plánovacího manažera můžeme zvolit jakýkoli počítač v síti k tomu nastavený, stačí znát jeho IP adresu nebo jeho jméno. V tomto případě nastavíme manažera na vlastní počítač zadáním jména svého počítače nebo IP adresy, trial verze Test Studia nám nabízí možnost nastavit až sto virtuálních uživatelů zatěžujících naše webové stránky. Zátěž, kterou budeme testovat naše stránky, můžeme vytvořit buď z některého z funkčních testů, nebo manuálně, v horní tabulce vybereme ikonu pro zaznamenávání zátěže, v menu zvolíme, jestli chceme zátěž vytvořit manuálně nebo pomocí webového testu, poté vybereme prohlížeč, který bude zátěž zachycovat, popřípadě jestli chceme sledovat všechny lokální nebo vzdálené požadavky, poté se samo spustí nahrávání zátěže. Při manuálním nahrávání se otevře webový prohlížeč a my procházíme stránky tak, jak chceme, aby se choval virtuální uživatel, v případě použití webového testu se uživatel bude chovat jako zvolený test. Po nahrání můžeme ještě upravit zachycený provoz sítě a zátěž pojmenovat. Jsou-li potřeba identifikační údaje, je možné je přednastavit jako identitu. Také můžeme sledovat zatížení počítače, na kterém zátěžový test poběží, to nastavíme ve spodní tabulce přidáním nového monitoru výkonu, v našem případě je možné nastavit vlastní počítač. Dalším krokem je nastavení průběhu testu, zvolit počet virtuálních uživatelů, čas vykonávání testu a rozložení uživatelů v tomto čase, tímto je zátěžový test připraven ke spuštění. Těsně před samotným spuštěním testu můžeme nastavit, jak často chceme vzorkovat komunikaci. Dále, můžeme nastavit cíle testu, to jsou podmínky, které má test splnit, aby mohl být označen jako úspěšný. Je tu již několik přednastavených možností, ale je možné vytvořit cíl vlastní. Poté už jen test spustit a čekat na výsledky.

6.1.8. Testy výkonu

Výkonové testy je možno vytvořit přímo z automatických testů funkčních a fungujících, které nejsou spojeny s testovacími daty. V případě testů, které jsou propojeny s testovacími daty, skončí výkonový test chybou a nepřidá žádné statistiky do výsledků testu. Pro spuštění výkonového testu otevřeme již vytvořený funkční webový test a v horním navigačním menu přejdeme do záložky „Performance“. Zde klikneme na „Configure“ a nastavíme úložiště výsledků testu a monitorovaný počítač. Test je připraven k spuštění, po spuštění proběhne námi vybraný webový test a zobrazí se výsledky výkonového testu, které nám zobrazí odezvu klienta i serveru a jejich poměr, v detailnějším pohledu můžeme zjistit všechny podrobnosti, které při testu proběhly.

6.1.9. Plánování testů

Nástroj Test Studio umožňuje testy v rámci projektu skládat do testovacích listů a tyto listy plánovat. Pro vytvoření listu testů přejdeme do záložky „Test Lists“, kde můžeme vytvářet listy manuálně nebo dynamické soubory testů, ty se filtrují podle pravidel, jako je třeba jméno testu, jeho vlastník nebo popis. Po vytvoření listu testů

můžeme naplánovat den, čas a jeho opakování, pod položkou „Schedule TestList“ nebo celý list nechat rovnou vykonat tlačítkem „Run List“.

6.1.10. Výsledky testů

Výsledky a naplánované testy si lze prohlédnout v záložce „Results“ projektu, zde jsou zobrazeny všechny testy, které proběhly nebo jsou naplánované, v přehledné časové linii. Lze si prohlédnout každý test, také kde v testu došlo k chybě a jak test uspěl. Záložka „Reports“ nabízí pohled na všechny testy a zobrazuje jejich úspěšnost ve sloupcovém grafu podle časové řady, kdy byly testy provedeny. Všechny informace lze přehledně třídit a prohlížet. Pro hlášení chyb podporuje Test Studio další nástroj od společnosti Telerik a tím je TeamPulse [38], který se stará o správu projektů.

6.2. Ukázkový případ použití nástroje Jubula

K ukázkovém případu pro nástroj Jubula jsem jako testovanou aplikaci zvolil Java File Managera, protože program obsahuje širokou škálu grafických komponent s kterými Jubula umí pracovat. Program je napsaný v jazyce Java za použití knihovny Swing, což je knihovna pro tvorbu uživatelského rozhraní pro jazyk Java. K ukázkovému případu jsem přiložil i vyexportovaný projekt z nástroje Jubula a excelovský soubor se vstupními daty pro vytvořený test.

6.2.1. Instalace

Instalační balíček lze stáhnout z domovských stránek [23]. Pro stažení Jubuly je požadována registrace na domovských stránkách nástroje. Po stažení spustíme instalační balíček a řídíme se podle instrukcí instalátoru. Po instalaci spustíme pomocí ikony na ploše nebo v nabídce start popřípadě v nabídce Metro u Windows 8.

6.2.2. První spuštění

Při prvním spuštění se nástroj zeptá na domovský adresář, kam bude ukládat všechna data, adresář zvolíme podle svého uvážení. Následně nás přivítá uvítací stránka, na které je výběr z několika sekcí. Vybereme ikonu s popiskem „Workbench“ a uvítá nás základní pracovní plocha.

6.2.3. Vytvoření projektu

Pro vytvoření nového testovacího projektu zvolíme položku Test z horní navigační lišty. Objeví se tabulka pro vytvoření nového projektu, jako jméno projektu zvolíme název našeho projektu, v rolovacím výběru „Toolkit for Test Specification“ vybereme položku Swing. Dále můžeme vybrat jazyk pro testovací data, pro tuto ukázku necháme ve výběru angličtinu a pokračujeme na další krok. V dalším kroku vyplníme jméno pro „Application Under Test“ (AUT) agenta a jako sadu nástrojů zvolíme Swing. Následující krok slouží k nastavení AUT, pod tlačítkem „Add“ vybereme cestu k naší testované aplikaci. Vybrat můžeme spouštěcí dávkový soubor nebo spustitelný soubor aplikace. Pro správné chování AUT je nutné nastavit cestu k souboru *java.jar* uloženém v místě instalace prostředí Java. Při úspěšném vytvoření projektu vidíme v sekci „Test Suite Browser“, na levé straně, že nám přibyl projekt.

6.2.4. Vytvoření testu

Vlevo dole se nachází sekce s názvem „Test Case Browser“, ve které pravým tlačítkem klikneme na položku „Test Cases“, pod nabídkou „New“ vybereme „New Test Case“ a zvolíme jméno testovacího případu. Po rozkliknutí položky „Test Cases“ vidíme další položku s názvem „unbound_modules_concrete_[7.2]“ a námi nově vytvořený test, který otevřeme poklikáním myší.

Uprostřed pracovní plochy se otevře záložka s názvem našeho testu, v tuto chvíli můžeme začít přidávat akce z položky „unbound_modules_concrete_[7.2]“. Vybranou akci přetažením přidáme do testovacího případu a v pravém horním rohu v tabulce vyplníme potřebné nastavení akce. U akcí vyplníme jméno akce, parametry akce, ty můžeme nastavit jako pevné přímo v nastavení akce, nebo je nastavit jako proměnnou, zapsáním znaku rovnítka a názvu proměnné, která bude do testu vstupovat. Všechny vstupní parametry jsou vypsány v názvu testu v hranatých závorkách. Jako ukázku jsem vytvořil test pro funkční testování GUI pro souborový manažer napsaný v jazyce Java za použití knihovny Swing.

Zde uvedu jednotlivé kroky ukázkového případu, v jaké podsložce se akce nachází a její jméno, nastavení parametrů uvedu v tabulce 10: Tabulka akcí a jejich parametrů, která je přiložena v příloze:

1. Zjištění, jestli je okno aplikace viditelné
    - Check>Application, ub_app_checkExistenceOfWindow
2. Otevření konfigurací z horního navigačního menu.
    - Select>Menu Bar, ub_mbr_selectEntry_byTexpath
3. Vybrání uzlu „Fonts“ ze stromu nabídky.
    - Select>Tree, ub_tre_selectNode_byTextpath
4. Změna fontu na některý z nabídky.
    - Select>List, ub_lst_selectEntry_byValue
5. Vytvoření a pojmenování složky pomocí tlačítka MKDIR.
    - Input via Keyboard>Componetn with Text Input, ub_cti_replaceText
6. Vyhledání a vstup do vytvořené složky.
    - Select>Table, ub_tbl_selectValueFromComlumn
7. Restartování aplikace
    - Restart Application, >Restart, ub_app_restart

Ke každému kroku se vztahuje klinutí myší na některé tlačítko, před vytvářením doporučuji si kroky projít manuálně a vyzkoušet si, jak má celý postup vypadat, postupně přidávat akci pro kliknutí myší ub_grc_clickLeft_single uloženou v podsložce Click.

Tuto sekvenci akcí jsem vybral, protože dobře ukáže možnosti práce s nástrojem Jubula, a při vykonání všech kroků narazíte na různorodé grafické komponenty a praci s těmito komponentami.

Při vkládání jednotlivých akcí do seznamu je možné pojmenovat jednotlivé akce v sekci „Coponent Names“, tyto názvy se pak zobrazí při mapování grafického prostředí a je důležité vědět, ke které akci jakou grafickou komponentu přidat, právě k tomu slouží pojmenování.

Po vytvoření testu se všemi akcemi, které jsou uvedené v tabulce 10: Tabulka akcí a jejich parametrů v příloze, je ještě třeba vytvořit vstupní data pro test, což se provede v sekci „Data Sets“, kde jsou již připravené sloupce s názvem proměnných, přidání provedeme tlačítkem „Add“ nebo můžeme importovat data z centrální databáze.

Posledním krokem je přidání testu do testovacího souboru testů, ten vytvoříme kliknutím pravým tlačítkem myši na název projektu v sekci „Test Suite Browser“, vybráním položky „New“ a zvolením „New Test Suite“. Takto vytvořený soubor rozklineme a přetažením do něj přidáme náš test ze sekce „Test Case Browser“. V tuto chvíli je test připraven na mapování grafického rozhraní testované aplikace.

6.2.5. Vstupní data

Pro vytvoření vstupních dat klikneme na název testu a v pravém dolním rohu na záložku „Data Sets“, zde jsou zobrazeny všechny parametry, u kterých je možné přidat vstupní data, klikneme na tlačítko „Add“ a přidáme jeden z možných názvů fontů, které testovaná aplikace podporuje například Arial. Další proměnnou je název složky, kterou budeme vytvářet, a poslední je název složky, kterou budeme v souborovém manažeru hledat a otevírat, pro ukázkové účely doporučuji zadat oba názvy složek alespoň jednou stejné, aby mohl test proběhnout v pořádku.

Pro přidání dat do centrální databáze slouží ikona v horním navigačním menu s popisem „Central Test Data Editor“, pod, kterou můžeme vytvořit nový set dat kliknutím pravým tlačítkem myši a vybráním vytvoření nového setu dat. Při vytváření musíme zadat název nového datového setu, po vytvoření na set poklikáme a zobrazí se tabulka, kde pojmenujeme proměnné a určíme jejich datové typy, poté data zadáme v sekci „Data Sets“ v pravém dolním rohu. Data lze importovat i pomocí excelovského souboru a to tak, že z navigačního menu vybereme položku „File“ a zvolíme položku „Import“, kde ze složky „Test“ vybereme „Central Data Test Sets“ a uvedeme cestu k zvolenému souboru. Nástroj zpracovává pouze soubory s příponou xls. Při vytváření souboru je důležité dodržet formát souboru. A to tak, že list s daty pojmenujeme podle jazyka, ve kterém jsou vytvořeny a který projekt podporuje, například en_UK, en_US, cz_CZ, de_DE a do prvního řádku tabulky vypsat všechny názvy proměnných.

6.2.6. Mapování grafického rozhraní

Po vytvoření testovací sekvence akcí, které má test provést, je ještě potřeba namapovat grafické rozhraní aplikace. Pro mapování budeme potřebovat spustit AUT agenta, který zprostředkovává spojení s testovanou aplikací a nástrojem Jubula. Agenta spustíme ikonou z plochy nebo nabídky Start popřípadě z nabídky Metro u Windows 8. Agenta necháme spuštěného a v nástroji připojíme pomocí ikony s popisem „Connect to AUT Agent“ v horní liště, dále spustíme samotnou aplikaci pomocí ikony, která je také v horní liště a má popis „Run AUT“.

Dalším krokem je otevření mapovacího editoru, to provedeme přes pravý klik na námi vytvořený soubor testů a z nabídky „Open with“ vybereme „Object Mapping Editor“. Otevře se nám nová záložka na hlavní pracovní ploše. V této záložce jsou na levé straně názvy jednotlivých akcí, jak jsme je pojmenovali v „Component Names“. Kliknutím na ikonu „Start Object Mapping Mode“ v horní liště začneme nahrávání, což poznáme podle toho, že objekty ve spuštěné aplikaci jsou ohraničeny zeleným ohraničením. Pro namapování objekt označíme a stiskneme klávesovou zkratku „Control+Shift+Q“, jež přidá do pravého sloupce objekty namapované z aplikace. Jednoduchým přetažením, k sobě náležících objektů z pravého do levého sloupce objekty spárujeme. Spárované objekty se zobrazí ve spodní tabulce. Je možné přidat k jednomu objektu z aplikace více akcí z testu. Posledním krokem je vypnutí mapovacího módu ikonou v horní liště.

6.2.7. Spuštění testu

Je-li test vytvořený, objekty aplikace namapované a vstupní data přidána, tak je test připraven ke spuštění. Spuštění se provádí ikonou „Start Test Execution“ v sekci „Test Suite Browser“. Je důležité, aby byl spuštěný AUT agent a pomocí něj i testovaná aplikace. Při průběhu testu je nutné, aby nebylo dále s počítačem manipulováno do doby, než se test provede do konce, jinak test do konce neproběhne a neprovede všechny kroky testu.

6.2.8. Výsledek testů

Po úspěšném nebo neúspěšném vykonání všech testů nám nástroj ukáže vyhodnocení průběhu testu a přitom se přepne do pohledu pro spuštění a testů. Pohledy mezi tvorbou testů a jejich vykonáváním přepínáme v pravém horním rohu.

6.3. Ukázkový případ použití nástroje Randoop

Randoop je nástroj pro automatické generováni JUnit testů. K ukázkovému případu je přiloženo několik zdrojových souborů s jednoduchými metodami, deadlockem, simulace producenta a konzumenta. V zdrojovém souboru TestPack.java jsou mimo jiné i čtyři metody u kterých má Randoop vygenerovat testy detekující chybu. Detekci deadlocku se mi Randoopem nepodařilo prokázat. K nástroji Randoop jsem nenatočil žádné ukázkové video protože, jeho ovládání je založené na textových příkazech pro příkazovou řádku.

6.3.1. Instalace

Nástroj Randoop funguje buď jako samostatný program, nebo jako plug-in do vývojového prostředí Eclipse. Spustitelnou distribuci nástroje můžete stáhnout na stránkách:
https://code.google.com/p/randoop/downloads/detail?name=randoop.1.3.4.jar a začít rovnou používat. Instalace do prostředí Eclipse (doporučuji mít k dispozici nejnovější verzi 4.3.2) je také jednoduchá, po spuštění Eclipse najdete v navigační liště pod „Help“ položku „Install new software“ po kliknutí vložíte do formuláře adresu: *http://randoop.googlecode.com/hg/plugin.updateSite/*, zaškrtnete nástroj Randoop ve výběru níž a potvrďte formulář. Tímto jste nainstalovali plug-in nástroje Randoop do prostředí Eclipse.

6.3.2. Generování testů

Při použití vývojového prostředí Eclipse se testy generují tak, že v „Package Exploreru“ najdeme zdrojový kód testované aplikace, klikneme pravým tlačítkem myši a pod položkou „Run As“ vybereme „Randoop Test Input“. V menu, které se objeví, můžeme vybrat pro které třídy nebo přímo metody chceme testy generovat. Na další straně dialogu můžeme ovlivnit, kolik testů bude nástroj generovat nebo jak dlouho se budou testy generovat. Lze také nastavit, aby nástroj vygeneroval pouze zdrojové texty testů, které jsou úspěšné a nebo neúspěšné popřípadě obě dvě možnosti. Další položky nastavují maximální velikost testů, timeout vláken, jak často se má v testu použít hodnota null nebo počet testů na jeden zdrojový soubor. Pro ukázkový případ nastavíme použití hodnoty null na 0.5 a počet vygenerovaných testů na 100. Po vygenerování testy najdeme ve složce projektu po názvem test v balíčku Randoop. Soubor bez pořadového čísla defaultně generován s názvem „RandoopTest“, sdružuje ostatní soubory s testy do jednoho celku. Další soubory již obsahují vygenerované JUnit testy.

Při použití spustitelného balíčku nástroj spustíme přes příkazovou řádku, v příkazové řádce se přesuneme do složky, kde se nachází spustitelný balíček a zdrojové texty.Pak lze použít syntaxe *java –ea –classpath randoop.jar randoop.main.Main*

příkaz, vše do jednoho řádku. Pro generování testů slouží příkaz *–gentests*, pro přidání tříd pro testování *--testclass*, takže příkaz pro vygenerování testů k ukázkovému případu by vypadal takto *java –ea –classpath randoop.jar randoop.main.Main --gentests --testclass=TestPack1.java.* A nástroj vygeneruje testy přímo do složky, kde se nachází testované zdrojové soubory. Pro ukázkový případ použijeme příkaz *java –ea –classpath randoop.jar randoop.main.Main gentests --testlist=mojetirdy.txt --timeout=60 --inputlimit=100 --null-ratio=0.5.*

6.3.3. Seznam základních příkazů

Zde uvádím seznam základních a nejdůležitějších příkazů pro spustitelnou distribuci nástroje Randoop: [33]

- --testclass=*string* : Odkazuje na jméno třídy, která se má otestovat
- --classlist=*string* : Odkazuje na soubor s výpisem tříd které se mají otestovat. V souboru musí být každá třída na jednom řádku.
- --methodlist=*string* : Odkazuje na soubor s výpisem metod, které mají být otestovány. V souboru musí být každá metoda na jednom řádku.
- --randomseed=*int* : Náhodné číslo (seed) použité pro generovací proces [defaultně 0].
- --timelimit=*int* : Maximální počet sekund po, které se budou testy generovat [defaultně 100].
- --inputlimit=*int* : Maximální počet vygenerovaných testů. Použito pro zjištění kdy přestat testy generovat.
- --sting-maxlen=*int* : Maximální délka řetězců, které se budou při testování používat.
- --forbid-null=*boolean* : Nastavení zda se bude používat hodnota null jako vstup pro metodu nebo konstruktor.
- --null-ratio=*double* : Hodnota mezi 0 a 1, která určuje, jak často bude používána hodnota null jako parametr pro volání metod (1 znamená vždy volat s hodnotou null, 0 znamená nikdy nepoužívat hodnotu null).
- --output-tests=*string* : Rozhodnutí jaké testy nechat generovat úspěšné (pass), neúspěšné (fail) nebo všechny (all).
- help : Zobrazení nápovědy nástroje.

6.3.4. Spuštění testů

Pro spuštění vygenerovaných testů je potřeba nástroj JUnit, který je dostupný na stránkách *junit.org*. Knihovny nástroje JUnit jsou součástí vývojového prostředí Eclipse a do projektu je lze přidat tak, že na projekt kliknete pravým tlačítkem myši a zvolíte „Build Path“, kde zvolíte „Add Libraries“, poté zvolíte položku JUnit a necháte naimportovat knihovnu do projektu. Chcete-li použít spustitelný balíček nástroje JUnit, vložte balíček do cesty třídy (classpath). [39]

6.3.5. Výsledky testů

Nástroj Randoop generuje testy dvou druhů a to testy označující chybu, ty lze použít k vyhledání a odstranění chyby v kódu, a testy regresní, které zachycují chování programu, a lze jimi zjišťovat, jestli změny provedené ve zdrojovém textu nevytvořily chyby v již fungujícím zdrojovém textu. O vlastní výsledky testů se stará nástroj JUnit. Pro použité ukázkové zdrojové soubory by měl nástroj při dodržení parametrů pro spuštění vygenerovat minimálně čtyři chybové testy.

Zde uvedu ukázku vygenerovaného testu nástroje Randoop, ukázka je ze zdrojového souboru, který jsem vytvořil pro testovací účely, jedná se o metodu, která násobí dvě celá čísla:

```
public void test9() throws Throwable {

    if (debug) System.out.printf("%nRandoopTest2.test9");


    int var2 = testExamples.TestPack1.multiplication(0, 100);//Spuštění
metody pro dělení dvou čísel.

    // Regression assertion (captures the current behavior of the code)
    assertTrue(var2 == 0);//Vytvoření pravidla pro zachycení výsledku.

        }
```

7. Ověření srozumitelnosti ukázkových případů

Pro ověření srozumitelnosti připravených ukázkových případů jsem provedl průzkum mezi dobrovolníky. Oslovil jsem převážně studenty stále studující na katedře informatiky a výpočetní techniky (KIV) při Západočeské univerzitě v Plzni. Podařilo se mi oslovit několik studentů studující obor Informační systémy a obor Informatika. Prozkoumání ukázkových případů mi přislíbilo 8 oslovených. Dobrovolníkům jsem zaslal ukázkové případy a další soubory potřebné k praktickému vyzkoušení ukázkových případů, a odkaz na formulář s otázkami. Ukázková videa jsem umístil na server Youtube jako neveřejná a dobrovolníkům zaslal odkazy.

7.1. Dotazník

Pomocí dotazníku chci ukázat, že vytvořené ukázkové případy jsou srozumitelné a dostatečně obsáhlé. Další otázky jsem volil otázky, abych zjistil zda účastníci ukázkové případy skutečně vyzkoušeli a který nástroj se jim líbil nejvíce. Oslovení dobrovolníci si měli přečíst a vyzkoušet ukázkové případy, poté měli odpovědět na několik otázek. Dotazník jsem vytvořil za pomoci služby Drive od společnosti Google a odkaz na stránku s dotazníkem jsem zaslal všem dobrovolníkům.

V dotazníku jsem se ptal na otázky:

- Byly ukázkové případy srozumitelné?
- Připadají Vám ukázkové případy dostatečné?
- Zhlédli jste ukázková videa?
- Byla videa srozumitelná?
- Vyzkoušeli jste si ukázkové případy prakticky?
- Co jste použili jako hlavní pomůcku?
- Museli jste při zkoušení případů hledat informace někde jinde než v PDF nebo videu? Případně, kde jste informace hledali?
- Jak byste změnili ukázkové případy?
- Který z nástrojů se Vám líbil nejvíce?
- Měli jste možnost pracovat s některým z nástrojů již dříve? Pokud ano, s kterými?

7.2. Výsledky a odpovědi

Z celkových 8 oslovených, kteří přislíbili prozkoumání ukázkových případů nakonec dotazník zodpověděli čtyři dobrovolníci, což je 50% všech, kterým byly ukázkové případy poslány. Všichni odpověděli, že ukázkové případy jsou srozumitelné. Všichni také odpověděli, že případy jsou dostatečné, ale až příliš podrobné. Všichni z dotazovaných také odpověděli, že ukázková videa zhlédli a že videa byla dostatečně srozumitelná. Pouze 50% všech, kteří dotazník vyplnili, uvedlo, že si vyzkoušeli všechny ukázkové případy prakticky, zbylých 50% si vyzkoušelo prakticky jen některé z nástrojů. Stejné rozdělení odpovědí se nacházelo i u další otázky a to, jestli jako hlavní pomůcku při zkoušení případů použili ukázkové případy ve formě textové nebo jako ukázková videa. Žádný dobrovolník nehledal další informace jinde než v textových případech nebo ve videích. Nejčastějším důvodem pro nevyzkoušení všech ukázkových případů byla jejich časová náročnost. Jedním z nejčastějších návrhů na změnu ukázkových případů bylo přidat i obrázky obrazovek při práci s nástroji, další navrhovali lépe rozčlenit text nebo změnit strohý styl psaní. Nástroj, který se dotazovaným nejvíce líbil, byl Test Studio s 50% z celkového počtu, zbylé dva nástroje

se rozdělili rovným dílem, tedy 25% každý. Všichni dotazovaní také odpověděli, že nikdy neměli zkušenost s některým z vybraných nástrojů. Přesné odpovědi jsou k nahlédnutí v tabulce 11: Tabulka výsledků dotazníku v příloze.

8. Závěr

Tématem práce byl „Přehled nástrojů pro automatické testování aplikací“ a jejím cílem bylo přiblížit problematiku automatizace testování, prozkoumání trhu s nástroji pro automatické testování a vytvoření ukázkových případů k několika vybraným nástrojům.

V teoretické části práce jsou uvedeny metodiky pro vývoj a testování softwaru. Věnuji se zde tomu, co je testování, co je to chyba a jaká je její cena. Prozkoumal jsem i co je to kvalita software, kde jsem se řídil podle normy ISO/IEC 9126 -1. Jednou z důležitých součástí vývoje softwaru jsou modely životních cyklů vývoje softwaru – spirálový model, přírůstkový model a některé z agilních metodik – snažil jsem se alespoň částečně popsat jejich princip.

V další části jsem prohledal trh s nástroji, hledal jsem i mimo komerční trh a přidal několik nezávislých nástrojů pro automatické testování. Vytvořil jsem přehled několika zajímavých nástrojů, zajímal jsem se hlavě o jejich použití, jakou technologii podporují, jak automatizují, jejich cenu a podporu od vývojářů nebo od komunity, která nástroj používá. Poté jsem vybral tři nástroje tak aby každý ukazoval jiné možnosti automatizace. Konkrétně jsem vybral: Test Studio, Jubulu a Randoop.

Test Studio jsem vybral jako zástupce testování webových stránek a aplikací. Nástroj je robustním řešením pro vývojové týmy a podle mého je to nástroj vhodný pro střední a velké firmy zabývající se vytvářením webových aplikací. Podává pomocnou ruku při funkčním, zátěžovém a výkonovém testování, vše jednoduše a přehledně.

Jubula je nástrojem pro funkční testování grafického rozhraní. Takovýto nástroj je užitečný při vytváření jak malých tak velkých aplikací které disponují grafickým rozhraním. Je to nástroj určený pro práci s aplikacemi vytvořenými v jazyce Java.

Randoop svojí rychlostí při generování jednotkových testů je využitelný téměř v každé fázi vývoje nějakého softwarového produktu. Umí rychle říci, zda vytvořený zdrojový text neobsahuje vážné chyby a je-li zachována funkčnost již napsaných částí.

K těmto třem nástrojů jsem vytvořil ukázkové případy použití, ve kterých jsem popsal použití nástrojů, instalaci, nastavení nástroje, vytvoření testů, spuštění a plánování testů a případné vyhodnocení testů. Ukázkové případy jsem psal podrobně, aby je mohl použít i někdo, kdo s nimi nemá žádnou zkušenost. Pro lepší pochopení ukázkových případů jsem vytvořil několik videí, které ukazují jak postupovat při vytváření ukázkových případů. Poslední částí práce bylo zjistit, zda jsou ukázkové případy srozumitelné a vhodně zvolené. Pro ověření jsem oslovil několik dobrovolníků a předložil jim ukázkové případy k vyzkoušení. Dobrovolníci měli ukázkové případy prakticky vyzkoušet a vyplnit dotazník. V dotazníku všichni dobrovolníci označili ukázkové případy za srozumitelné a dostatečné, pouze upozornili, že ukázkové případy jsou až příliš podrobné, poukázali na strohý styl psaní a doporučili přidat obrázky s postupem práce.

Jako hlavní přínos této práce bych označil obecné shrnutí základů testování, jeho automatizace a vytvoření přehledu některých testovacích nástrojů. Dále vytvoření srozumitelných ukázkových případů pro několik nástrojů pro automatické testování a jejich vizualizace v podobě videí.

Přehled pojmů a zkratek

AJAX – Technologie pro vývoj interaktivních webových stránek, které umí měnit obsah. [40]

API – Zkratka pro Application Programming Interface. Jedná se soubor metod, procedur, tříd nebo protokolů, které může programátor používat. [41]

AUT – Zkratka pro Application Under Test, jedná se o spojku mezi testovanou aplikací a testovacím nástrojem. Používáno u nástroje Jubula. [24]

C# - Objektově orientovaný programovací jazyk založený na jazycích C++ a Java vyvinutý společností Microsoft. [40]

CSS – Kaskádovité styly, jazyk pro popis vzhledu webových stránek při použití jazyků HTML. [40]

CSV – Souborový formát pro výměnu tabulkových dat. Jednotlivé hodnoty v řádku jsou odděleny čárkou. [42]

drag&drop – V překladu jako „táhni a pusť“ jedná se o gesto kdy myší přesouváme vybraný objekt z místa A do místa B. [43]

EULA – End-User Licence Agreeement, je typ softwarové licence pro koncového uživatele. [44]

GmbH – Německá zkratka pro společnost s ručením omezeným.

GUI – Graphical User Interface je zkratka pro grafické rozhraní aplikace.

HTML – HyperText Markup Language je značkovací jazyk pro vytváření webových stránek. [40]

IBM – International Business Machines Corporation je akciová společnost, která se zabývá oborem informačních technologií. [40]

IEC – International Electrotechnical Commission je organizace, která publikuje a vypracovává mezinárodní normy pro obory elektrotechniky, elektroniky, sdělovací techniky a příbuzných oborů. [41]

iOS – Je uzavřený mobilní operační systém vytvořený společností Apple určený pouze pro produkty této společnosti. [45]

IP – Je internetovým protokolem používaným v počítačových sítích na síťové vrstvě, který poskytuje diagramové služby. Využívané jsou verze IPv4 a IPv6. [46]

jar – Souborový formát používaný platformou Java pro distribuci programů.

JUnit – Knihovna pro jednotkové testy v jazyce Java. [39]

MIT – Massachusetts Institute of Technology je soukromá výzkumná univerzita ve městě Cambridge, v americkém státě Massachusetts. [41]

MySQL – Je multiplatformní databázový systém. [40]

Pearl – Je interpretovaní programovací jazyk, který se stal velmi populárním pro psaní skriptů pro webové stránky.

PHP – Hypertext Preprocessor, původně Personal Home Page je skriptovací programovací jazyk pro vytváření dynamických internetových stránek. [41]

plug-in – V překladu „zásuvný modul“ je software, který nepracuje samostatně, ale funguje jako doplňkový modul pro jiné aplikace. [40]

QML – Conformiq Modelling Language je modelovací jazyk vytvořený firmou Conformiq pro vytváření modelů systému. [17]

RCP – Rich Client Platform jsou nástroje pomocí kterých je možné rychle vytvořit prostředí aplikace. [47]

Ruby – Je interpretovaný skriptovací programovací jazyk, je objektově orientovaný. [40]

Swing – Je knihovna grafického rozhraní pro programovací jazyk Java, pomocí knihovny Swing lze vytvářet okna, dialogy, tlačítka a další graficky uživatelské rozhraní. [48]

SWT – Standard Widget Toolkit je knihovna grafických uživatelských prvků pro programovací jazyk Java. [49]

UML – Unified Modelling Languageje grafický jazyk pro vizualizaci, návrhy a dokumentaci programových systémů. [41]

URL – Uniform Resource Locator je definovanou strukturou znaků podle kterých lze specifikovat umístění zdrojů informací. [40]

Workflow – V překladu „pracovní postup“ je schéma nějaké prováděné činnosti či procesu složené z jednotlivých kroků. [41]

XAML – Extensible Application Markup Language je značkovací jazyk, který se používá pro vytváření formulářů a vzhled uživatelské aplikace. [41]

xls – Je souborový formát používaný tabulkovým procesorem Excel od firmy Microsoft. [40]

XML – Extensible Markup Language je značkovací jazyk používaný pro serializaci dat. [40]

Youtube – Je internetový server pro sdílení videí, který vlastní společnost Google. [40]

Seznam tabulek a obrázků

Reference

[1] **Patton, Ron.** *Testování softwaru.* Praha : Computer Press, 2002. ISBN 80-7226-636-5.

[2]. Software Testing. *Wikipedia.* [Online] 17. duben 2014. [Citace: 20. duben 2014.] http://en.wikipedia.org/wiki/Software_testing.

[3]. **Vaníček, Jiří.** Vlastimil Čevela - osobní profil a realizované projekty/aktivity. *cev.cemotel.* [Online] 2009. [Citace: 15. duben 2014.] http://cev.cemotel.cz/programovani_a_tvorba_sw_1975-2004/2004/311.pdf.

[4]. **Havlíčková, Anna.** Testování Softwaru. [Online] 5. srpen 2008. [Citace: 11. duben 2014.] http://testovanisoftwaru.blogspot.cz/p/dptestovanisoftwarupdf.html.

[5]. Difference between defect, error, bug, failure and fault. *Tfortesting.* [Online] 3. září 2012. [Citace: 7. květen 2014.] http://tfortesting.wordpress.com/2012/09/03/difference-between-defect-error-bug-failure-and-fault/.

[6]. **Sommeville, Ian.** *Software engineering.* New York : Addison-Wesley, 2001. Sv. 6th. ISBN 0 201 39815 X.

[7]. Software development technology. *Wikipedia.* [Online] 6. duben 2014. [Citace: 12. duben 2014.] http://en.wikipedia.org/wiki/Software_development_methodology.

[8]. Barry Boehm. *Wikipedia.* [Online] 25. duben 2014. [Citace: 30. duben 2014.] http://en.wikipedia.org/wiki/Barry_Boehm.

[9]. Metodika vývoje softaru. *Wikipedia.* [Online] 17. duben 2014. [Citace: 5. květen 2014.] http://cs.wikipedia.org/wiki/Metodika_v%C3%BDvoje_softwaru.

[10]. **Schach, Stephen R.** *Software Engineering.* Homewood, IL : Irwin, 1990. ISBN 02-560-8515-3.

[11]. Harlan Mills. *Wikipedia.* [Online] 24. prosinec 2013. [Citace: 12. březen 2014.]

[12]. **Pressman, Roger S.** *Software Engineering.* New York : McGraw - Hill, 2001. ISBN 0073655783.

[13]. Agile software development. *Wikipedia.* [Online] 29. duben 2014. [Citace: 5. květen 2014.] http://en.wikipedia.org/wiki/Agile_software_development.

[14]. Extreme programming. *Cunningham & Cunningham, Inc.* [Online] 28. říjen 2013. [Citace: 5. květen 2014.] http://c2.com/cgi/wiki?ExtremeProgramming.

[15]. What is scrum? *Scrum.org.* [Online] 2014. [Citace: 5. květen 2014.] https://www.scrum.org/resources/what-is-scrum/.

[16]. Conformiq Modeler. *Conformiq.* [Online] [Citace: 1. květen 2014.] http://www.conformiq.com/products/conformiq-modeler/.

[17]. Conformiq Manual. *Verifysoft.* [Online] 27. březen 2014. [Citace: 5. květen 2014.] http://www.verifysoft.com/ConformiqManual.pdf.

[18]. Conformiq Designer. *Conformiq.* [Online] 2014. [Citace: 1. duben 2014.] http://www.conformiq.com/products/conformiq-designer/.

[19]. Mobile Testing. *Telerik.* [Online] Telerik. [Citace: 25. duben 2014.]

[20]. Test Studio Forum. *Telerik.* [Online] Telerik. [Citace: 25. duben 2014.] http://www.telerik.com/forums/teststudio.

[21]. Test Studio. *Telerik.* [Online] Telerik, 2014. [Citace: 12. březen 2014.] http://www.telerik.com/teststudio.

[22]. Peta-testautomation. *Peta.* [Online] Verit, 2012. [Citace: 20. březen 2014.] http://www.peta-testautomation.com/home.

[23]. GUIdancer&Jubula. *Bredex.* [Online] Bredex GmbH, 2014. [Citace: 2. duben 2014.] http://www.bredex.de/guidancer_jubula_en.html.

[24]. Jubula Support. *Eclipse.* [Online] The Eclipse Foundation, 2014. [Citace: 28. březen 2014.] https://www.eclipse.org/jubula/support.php.

[25]. Agitar technologies. [Online] 2013. [Citace: 16. březen 2014.] http://www.agitar.com/.

[26]. Automatic Test Generation. *riedquat.* [Online] 26. září 2008. [Citace: 15. březen 2014.] http://www.riedquat.de/blog/2006-12-03-01.

[27]. Randoop Projec Home. *Randoop.* [Online] 2013. [Citace: 2. duben 2014.] https://code.google.com/p/randoop/.

[28]. Concordion. [Online] 2013. [Citace: 13. duben 2014.] http://concordion.org/.

[29]. Company. *Telerik.* [Online] Telerik, 2014. [Citace: 12. březen 2014.] http://www.telerik.com/company.

[30]. Telerik UI for ASP.NET AJAX Documentation. *Telerik.* [Online] Telerik, 2014. [Citace: 25. duben 2014.] http://www.telerik.com/help/aspnet-ajax/introduction.html.

[31]. Automating Eclipse Jubula Tests with Jenkins. *Development Notes.* [Online] 14. červen 2011. [Citace: 5. květen 2014.] http://devnotesblog.wordpress.com/2011/06/14/automating-eclipse-jubula-tests-with-jenkins/.

[32]. Meet Jenkins. *Jenkins.* [Online] 7. leden 2014. [Citace: 5. květen 2014.] https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins.

[33]. Randoop Manual. *Randoop.* [Online] 2013. [Citace: 3. duben 2014.] http://randoop.googlecode.com/hg/doc/index.html.

[34]. Java File Manager. *Sourceforge.* [Online] 14. červenec 2006. [Citace: 22. duben 2014.] http://jfm.sourceforge.net/.

[35]. Deadlock. *The Java tutorials.* [Online] 2014. [Citace: 12. duben 2014.] http://docs.oracle.com/javase/tutorial/essential/concurrency/deadlock.html.

[36.] CS349. *Welleslay college.* [Online] 2002. [Citace: 12. duben 2014.]

[37]. Xampp. *Sourceforge.* [Online] 2014. [Citace: 15. Březen 2014.]

[38]. TeamPulse. *Telerik.* [Online] Telerik, 2014. [Citace: 25. duben 2014.] http://www.telerik.com/teampulse.

[39]. Download and Instal. *Github.* [Online] 2014. [Citace: 1. květen 2014.] https://github.com/junit-team/junit/wiki/Download-and-Install.

[40]. Počítačový slovník. *IT Slovník.* [Online] 2014. [Citace: 7. květen 2014.] http://it-slovnik.cz/.

[41]. **Říha, RNDr. Petr a Klaška, ing. Luboš.** Slovník počítačové informatiky a sítí. *Svět sítí.* [Online] 2014. [Citace: 7. květen 2014.] http://www.svetsiti.cz/slovnik.asp.

[42]. Comma-separated values. *Wikipedia.* [Online] 22. duben 2014. [Citace: 7. květen 2014.] http://en.wikipedia.org/wiki/Comma-separated_values.

[43]. Co znamená drag and drop? *1. PC revue.* [Online] 21. červen 2000. [Citace: 7. květen 2014.] http://www.1pcrevue.cz/tip0276.htm.

[44]. End-user license agreement. *Wikipedia.* [Online] 6. duben 2014. [Citace: 7. květen 2014.] http://en.wikipedia.org/wiki/End-user_license_agreement.

[45]. iOS. *Wikipedia.* [Online] 5. květen 2014. [Citace: 7. květen 2014.] http://en.wikipedia.org/wiki/IOS.

[46]. Internet Protocol. *Wikipedia.* [Online] 8. květen 2014. [Citace: 8. květen 2014.] http://en.wikipedia.org/wiki/Internet_Protocol.

[47]. Rich client platform. *Wikipedia.* [Online] 5. květen 2014. [Citace: 7. květen 2014.] http://en.wikipedia.org/wiki/Rich_Client_Platform.

[48]. Swing (Java). *Wikipedia.* [Online] 24. únor 2014. [Citace: 7. květen 2014.] http://en.wikipedia.org/wiki/Swing_(Java).

[49]. Standard Widget Toolkit. *Wikipedia.* [Online] 3. květen 2014. [Citace: 7 . květen 2014.] http://en.wikipedia.org/wiki/Standard_Widget_Toolkit.

[50]. Winston W. Royce. *Wikipedia.* [Online] 26. duben 2014. [Citace: 10. březen 2014.] http://en.wikipedia.org/wiki/Winston_W._Royce.

[51]. Vyvíjíme databázový a informační systém II. *dbsvet.cz.* [Online] 12. květen 2004. [Citace: 5. květen 2014.] http://www.dbsvet.cz/view.php?cisloclanku=2004051201.

Přílohy

Příloha A – Tabulka akcí pro ukázkový případ nástroje Jubula

Název akce	Parametry	Nastavení parametrů
checkExistenceOfWindow	Title	Java File Manager
	Operator	matches
	Exists	true
selectEntry_byTextpath	Textpath	Configuration.*/Show Configuration.*
	Operator	matches
selectNode_byTextpath	Path_Type	absolute
	Pre_Ascend	0
	Textpath	Display.*/Fonts.*
	Operator	matches
	Number_Of_Clicks	1
	Mouse_Button	1
	Extende_Selection	no
clickLeft_Single		
selectEntry_byValue	Entry_Name	=FONTNAME
	Operator	matches
	Search_Type	absolute
	Extend_Selection	no
	Mouse_Button	1
	Number_Of_Clicks	1
clickLeft_Single		
replace_Text	Text	=MKDIRNAME
clickLeft_Single		
selectValueFromColumn	Column	Name.*
	Column_Operator	matches
	Value	=TABLEVALUE
	Number_Of_Clicks	2
	Extend_Selection	yes
	Search_Type	absolute
	Mouse_Button	1
restart		

Tabulka 10: Tabulka akcí a jejich parametrů

Příloha B – Tabulka výsledků dotazníku

Časová značka	28.4.2014 23:24:46	28.4.2014 23:41:55	29.4.2014 23:01:55	4.30.2014 9:33:44
1. Byli ukázkové příklady srozumitelné?	ANO	ANO	ANO	ANO
2. Připadají Vám ukázkové případy dostatečné?	ANO (Ale je to moc podrobné)	ANO (Ale je to moc podrobné)	ANO (Ale je to moc podrobné)	ANO (Ale je to moc podrobné)
3. Shlédli jste videa ?	ANO	ANO	ANO	ANO
4. Byla videa srozumitelná??	ANO	ANO	ANO	ANO
5. Vyzkoušeli jste si ukázkové případy prakticky?	Pouze některé nástroje	Pouze některé nástroje	ANO	ANO
6. Co jste použili jako hlavní pomůcku?	PDF	Video	PDF	Video
7. Museli jste při zkoušení případů hledat informace někde jinde než v PDF nebo videu? Kde?	Ne	Ne	ne	
8. Proč jste si případy nevyzkoušeli?		Časová náročnost		
9. Jak by jste změnili ukázkové případy?	- přidat obrázky do PDF - text by bylo vhodné zkrátit a lépe členit, některé detaily mi přijdou nadbytečné	Některé úkony jsou zbytečně rozepsány, vzhledem k tomu, že je návod určen pro programátory. Text by bylo vhodné více zpřehlednit například rozdělení odstavců a přidání obrázků (někdy by řekly více než text).	U návodů bych přidal screenshoty oken nástrojů - je to potom více názorné. (jen můj názor)	Myslím si, že jednotlivé případy jsou zvoleny poměrně vhodně. Pozměnil bych strohý styl psaní tak, aby alespoň trochu zaujal.

10. Který z nástrojů se Vám líbil nejvíce?	Randoop	Test Studio	Jubula	Test Studio
11. Měli jste možnost pracovat s některým z nástrojů již dříve? Pokud ano s kterými?	NE	NE	NE	NE

Tabulka 11: Tabulka výsledků dotazníku

Příloha C – Obsah přiloženého CD

Jubula:

- Testovaná aplikace JavaFileManager
  - jfm.jar
- Vyexportovaný projekt ukázkového případu FileManager_UkazkovyPripad_1.0.xml
- Vstupní data pro ukázkový případ jubulaDataInput.xls

Randoop:

- JUnit testy vygenerovaná nástrojem Randoop
  - Testy pro dreadlock
  - Testy pro ProducentKonzument
  - Testy pro TestPack
- Zdrojové texty testovaných programů
  - ProducentKonzument.java
  - TestDeadLock.java
  - TestPack.java
- Spustitelná distribuce nástroje Randoop randoop.jar
- Textový soubor s názvy tříd tridy.txt

Test Studio:

- Testovací webové stránky
  - Zdrojové texty webových stránek
- Soubory projektu ukázkového případu
  - Soubory vygenerovaná nástrojem TestStudio
- Testovací vstupní data TestStudioDataInput.xlsx

Ukázková videa pro nástroje Jubula a Test studio:

- Jubula_Videa
  - Videa ukazující postup vytvoření testovacích případů nástrojem Jubula
- TestStudio_Videa
  - Vide ukazující postup vytvoření testovacích případů nástrojem Test Studio

Dotazník:

- Automatizované nástroje pro testování aplikací (Odpovědi).pdf
- Dotazník_ Automatizované nástroje pro testování aplikací - Formuláře Google.pdf