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

Bakalářská práce

Přehled nástrojů pro automatické testování GUI

Plzeň 2016 Irena Drápalíková

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Richardu Lipkovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost během konzultací.

Prohlášení

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

V Plzni dne 5. května 2016

Irena Drápalíková

Abstrakt

Tato bakalářská práce poskytuje základní teoretické poznatky v oblasti testování grafických uživatelských rozhraní. Vysvětluje základní metody využívané pro toto testování. Dále se zabývá automatizací testování. Obsahem práce je přehledný seznam nástrojů, které jsou pro tento účel na současném trhu k dispozici. Součástí popisu každého nástroje je popis kvality jeho podpory, forma licence, pod kterou je nástroj šiřitelný a v neposlední řadě také přehled technologií, které nástroj podporuje. Následuje podrobnější porovnání nástrojů Jubula a SikuliX, protože se jedná o jediné nástroje, které jsou k dispozici zdarma a současně umožňují testovat aplikace napsané v jazyce Java. Tato podrobnější analýza obsahuje sedm vytvořených testů v každém nástroji. Během zpracovávání těchto testů byl také každý nástroj hodnocen a byly zaznamenávány nalezené výhody a nevýhody. V závěru práce je zhodnocena jejich uživatelská použitelnost. Toto zhodnocení je založeno na zkušenostech s vytvářením vzorových testů, na měření času běhu jednotlivých testů a na případové studii. Ta proběhla s pěti uživateli. Uživatelé plnili předepsané úlohy a hodnotili každý nástroj. Součástí případové studie je zejména měření času, který je potřeba pro vytvoření testu v konkrétním nástroji. Na základě těchto kritérií se vítězem porovnání se stal nástroj SikuliX.

Abstract

This bachelor thesis provides basic theoretical knowledge in graphical user interface testing. It explains the basic methods used for this testing. It also deals with testing automation. This work contains a list of tools that are available on the current market. Part of description of each tool is description of the quality of its support, a form of a license under which the tool is avaiable, and an overview of supported technologies. Followed by a detailed comparison of tools Jubula and SikuliX because they are the only tools that are available for free while allowing test applications written in Java. This detailed analysis includes seven tests developed in each software. During creating these tests software has also been evaluated and were registered advantages and disadvantages. In the conclusion is evaluation of user usability of Jubula and SikuliX. This evaluation is based on experience with creating tests, on time measuring course of individual tests and a case study. The case study was made with five users. Users created required tasks and evaluated each tool. Part of the case study is measuring the time it takes to create a test in each tool. Based on these criteria, the best is SikuliX.

Obsah

1 Úvod

Počítačový software je v dnešní době nepostradatelným pomocníkem v každodenním životě milionů lidí. Ať už se jedná o jeho využití v zaměstnání nebo soukromém životě, je očekávána jeho bezchybnost a stoprocentní funkčnost. Jelikož všechny aplikace vytvářejí lidé, nejsou chyby ničím neobvyklým. Proto je testování aplikace již přirozeným článkem jejího životního cyklu. Jak bylo zmíněno, v činostech, kde působí výhradně lidský faktor, jsou chyby běžné a problematicky ohalitelné, protože jsou obvykle v každém takovém případě jiné. Automatizované testování aplikací je tedy tím správným krokem ke zlepšení jejich kvality.

V první části této práce se věnuji obecným teoretickým poznatkům v oblasti testování. Dále provedu průzkum současného trhu s nástroji pro automatizované testování grafických uživatelských rozhraní a vytvořím přehledný souhrn nalezených nástrojů. Poté se budu věnovat několika konkrétním nejzajímavějším nástrojům a ty prozkoumám podrobněji. Následně předvedu jejich použití na konkrétních příkladech. V závěru se zaměřím na zhodnocení uživatelské použitelnosti těchto vybraných nástrojů.

Cílem práce není úplné vysvětlení problematiky testování grafických uživatelských rozhraní a jeho automatizace. Spíše slouží k vytvoření přehledu některých zajímavých nástrojů pro automatizované testování těchto rozhraní.

2 Uživatelská rozhraní

Uživatelská rozhraní jsou prostředníkem mezi uživatelem a strojem. Představují způsob, jakým lidé s určitým zařízením, v našem případě s počítačem, komunikují a jak ho ovládají. V minulosti byla hlavním ovládacím prvkem příkazová řádka. V dnešní době je pro běžného uživatele standardem ovládání počítače pomocí grafického uživatelského rozhraní.

2.1 Typy uživatelských rozhraní

2.1.1 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (anglicky Graphical User Interface, zkratkou GUI) je rozhraní, které umožnuje ovládat zařízení pomocí interaktivních grafických ovládacích prvků. Podrobněji v kapitole 2.2.

2.1.2 Příkazový řádek

Příkazový řádek (anglicky Comand Line Interface, zkratkou CLI) představuje uživatelské rozhraní, ve kterém uživatel komunikuje se systémem pomocí zapisování příkazů do příkazového řádku.

2.1.3 Textové uživatelské rozhraní

Textové uživatelské rozhraní (anglicky Text User Interface, zkratkou TUI) je rozhraní na pomezí GUI a CLI. V tomto rozhraní je obrazovka rozdělena na sloupce a řádky. V každé pozici je možné zobrazit pouze jeden znak z předem dané množiny, například ASCII.

2.1.4 Braillský řádek

Jedná se o kompenzační pomůcku, která umožňuje ovládání počítače nevidomým nebo těžce zrakově postiženým uživatelům. Převádí výstup počítače do Braillova písma. Často funguje také jako vstupní zařízení.

2.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (dále GUI) bylo vytvořeno za účelem usnadnění práce s počítačem. Komunikace mezi uživatelem a strojem probíhá pomocí

přímé manipulace s grafickými prvky, jako jsou ikony, menu, tlačítka, posuvníky. Takové rozhraní má podobu okna s těmito prvky, ve kterém uživatel zadává vstup a následně také čte výstup aplikace.

Umožňuje uživatelům pracovat s počítačem bez znalosti příkazů, jejichž znalost byla před používáním GUI nutnost. Také jeho výstup je uživatelsky přívětivější, zejména díky možnosti zobrazení více oken najednou. Uživatel tak vidí v jeden okamžik více informací a je schopen lépe porozumět aplikaci, se kterou pracuje. Je známo, že lidský mozek většiny lidí zpracovává lépe obrazové informace. Je schopen je zpracovat rychleji než text a také si obrazové vjemy lépe zapamatuje. S tím je spojena již zmiňovaná znalost příkazů. Díky GUI si uživatel lépe zapamatuje provádění určité činnosti, než kdyby se musel zpaměti naučit posloupnost konkrétních příkazů, které by vedly k požadované akci.

3 Testování GUI

Pod pojmem testování GUI si představme testování aplikace v podobě, v jaké s ní bude pracovat konečný uživatel. Tímto testováním předně kontrolujeme, zda aplikace funguje dle očekávání. Právě tyto testy je rozumné nějakým způsobem automatizovat. Dále se kontroluje GUI samotné. Jak už bylo řečeno, GUI zprostředkovává komunikaci aplikace s uživatelem, zobrazuje ovládací prvky, formuláře a výstupy. Kontrolujeme jejich zobrazení, funkčnost nebo zda správně probíhá požadovaná validace vstupů. Během testování by se nemělo zapomenout na různé chování aplikace nebo její zobrazování pro různé role uživatelů. V neposlední řadě se testuje také samotná použitelnost aplikace. Pro uživatele musí být přehledná, srozumitelná, jednoduchá. Všechny výše uvedené způsoby testování by měly být provedeny na různých strojích s různým softwarovým vybavením, aby byla testována také kompatibilita aplikace.

3.1 Důvody k testování GUI

Grafické uživatelské rozhraní je dnes již převažujícím způsobem komunikace mezi uživatelem a počítačem. Kvalitní GUI je jedním z klíčových faktorů, které ovlivňují úspěšnost a použitelnost systému, protože je to první věc, se kterou se uživatel během používání aplikace setká a s čím bude trávit celou dobu jejího využívání. Chyby, které se v GUI objeví, mohou v uživateli vyvolat dojem, že je celá aplikace chybná. Běžný uživatel neví, že onu chybu má na svědomí jen špatné zobrazení, ale vnitřně aplikace provedla vše tak, jak měla. Pro vývojáře je GUI prostředkem, který reprezentuje aplikaci, a samotná logika je ve vrstvě pod ním. Běžný uživatel má ale GUI na prvním místě, v případě chyby tedy oprávněně nabývá dojmu, že se jedná o chybu v celém softwaru. Proto by měl být na návrh rozhraní, jeho zhotovení a následné testování kladen velký důraz.

3.2 Typické chyby

V grafických uživatelských rozhraních se můžeme setkat s mnoha chybami. Některé jsou velmi časté, jiné by se daly označit až za raritní. Jednou z nejobvyklejších chyb je nedostatečné ošetření vstupů od uživatele. Jedná se o ošetření zadání správného datového typu zadávaného textu, nebo jeho formátu, ale také správné zpracování tohoto vstupu v programu. Velkým nedostatkem bývá také absence popisů tlačítek, formulářů a položek menu. Uživatel musí

mít přehled, jakou funkci objekty mají. Pokud již nějaká chyba v aplikaci nastane, je nutné, aby o ní byl uživatel informován srozumitelným a konkrétním hlášením. Oznámení o neočekávané chybě a následném ukončení programu uživateli příliš informací neposkytne. Dalším častým nedostatkem uživatelských rozhraní bývá absence tlačítek zpět a vpřed. Typickou chybou je nekonzistence aplikace. Aplikace někdy žádá o potvrzení akce a někdy ne, často podle toho, odkud je akce vyvolána. Má to obvykle na svědomí špatná struktura aplikace, kde se během každé takové akce volá jiná obslužná funkce. V GUI se také projevuje i celá řada chyb z nižších vrstev aplikace. Stačí jedna chyba v datovém modelu tabulky, například nefunkční přidávání dat do tabulky, a GUI nereaguje správně.

3.3 Metody testování GUI

3.3.1 Funkční a mimofunkční testy

Testy, které ověřují, že aplikace správně plní všechny úkoly pro které je určena, nazýváme funkční testy. Testují se obecně všechny funkce, které jsou v aplikaci implementovány, a ověřuje se, že fungují správně a že odpovídají požadavkům zákazníka.

Mimofunkční, někdy také označované jako nefunkční, testy spočívají v testování všech vlastností aplikace, které přímo nesouvisí s jejími funkcemi, ale zároveň jsou podstatné pro její správné fungování. Řadí se sem především výkonové testování, které má například ověřit, že aplikace bude pracovat dostatečně rychle i pod zátěží, jakou je větší počet současně pracujících uživatelů nebo větší objem zpracovávaných dat. Testuje se také připravenost aplikace pro budoucí nárůst zátěže. Toto testování by mělo zahrnout i vliv aplikace na stroj, na kterém běží. Tedy třeba to, zda zbytečně nezatěžuje paměť a procesor. K tomu se využívají benchmarky, které měří a porovnávají, kolik zdrojů aplikace v daném čase využívá.

Je tedy zřejmé, že funkční testy mají na výslednou kvalitu GUI významný vliv a popisu jejich druhů se budu věnovat níže. Proto je na tuto kategorii kladen veliký důraz během celého testovacího cyklu. Mimofunkční testování má také nepochybný vliv na výslednou bezporuchovost software. Bohužel v praxi na něj většinou není příliš pamatováno nebo se této kategorii testů nedostává dostatečná doba na provedení všech potřebných testů. [20] [21]

3.3.2 Testy dle znalosti vnitřní stuktury

V závislosti na znalosti či neznalosti programového kódu testované aplikace rozeznáváme tři typy testů. White box, gray box a black box, tedy testy bílé, šedé a černé skříňky.

U white box testů, někdy také nazývané testy průhledné skříňky, známe zdrojový kód aplikace. Tímto testováním můžeme lépe identifikovat možné problémy, protože známe přesně vnitřní strukturu programu a můžeme pak například testovat možné průchody kódem. To ovšem není jednoduchá záležitost, protože u rozsáhlejších aplikací existuje obrovské množství kombinací různých průchodů. Toto testování je důsledné, ale téměř nikdy se neztotožňuje s realistickým použitím aplikace. Testování touto formou s sebou také nese riziko, že tester test až příliš přizpůsobí činnosti programového kódu a nepodaří se mu software otestovat opravdu objektivně.

Během použití testů black box nemá tester ke zdrojovému kódu přístup, neví tedy, jakým způsobem testovaná aplikace došla k výsledku. Testovaný software v tomto případě představuje černou skříňku, kam tester zadá data a vidí pouze výstup, ověřuje se tak chování na rozhraní. Výhodou je, že tester v tomto případě nemusí být programátor a přinese nám tak jiný pohled na problém. Princip černé skříňky je pro testování GUI typický. Existuje i kombinace obou typů, ta se označuje jako testy gray box. V tomto případě například známe použitou datovou strukturu nebo algoritmus, ale samotnou implementaci neznáme. [1]

3.3.3 Testy dle přístupu

Provádějí se dva principiálně odlišné přístupy k testům, a to testy splněním a testy selháním. Testem splněním, často nazýváném jako test pozitivní, nazýváme takový test, kde v jeho průběhu zadáváme pouze vstupní hodnoty z množiny, kterou testovaná aplikace akceptuje. Nepokoušíme se záměrně vyvolávat situace, ve kterých očekáváme možnost výskytu chyby. Tento druh testů je obvyklý v počátečních fázích vývoje testované aplikace, kdy testujeme především její základní funkčnost. Jakmile se tester přesvědčí o funkčnosti aplikace za normálních okolností, je vhodné přistoupit k testům selháním, nazývaným také jako negativní testy. Během provádění testu selháním zadáváme pouze nekorektní hodnoty, které aplikace nepřijímá. Cílem tohoto testování je způsobení nečekaného ukončení programu. Výstupem by mělo být odpovídající chybové hlášení, nebo jiná reakce, která nevede k pádu programu. Negativní a pozitivní testy se používají velmi často a běžně se užívá

jejich kombinace. Většinou nejdříve proběhnou pozitivní testy a poté negativní. V praxi se často tyto testy provádějí v kombinaci s black box testy.

3.3.4 Testy progresní a regresní

Jak název napovídá, progresní testy využíváme pro kontrolu nových funkcí nebo vlastností aplikace. K jejich provedení tester potřebuje programátorskou dokumentaci, která přesně popisuje nově implementované funkce. Oproti tomu regresní testy slouží například k opakovanému otestování aplikace, zejména po opravení dříve nalezené chyby nebo otestování částí, které nebyly změněny doprogramováním nových vlastností. [20]

3.3.5 Dle zaměření

Tato oblast se také nazývá dimenze typu testů, nebo dimenze kvality. Tyto testy jsou často známy pod zkratkou FURPS+, kde jednotlivá písmena znamenají oblast, která je testována. Charakteristiky byly vytvořeny společností Hewlett-Packard.

F Funkčnost (Functionality)

U Použitelnost (Usability)

R Spolehlivost (Reliability)

P Výkon (Performance)

S Podporovatelonost (Supportability)

\+ Ostatní

Názvy oblastí již samy napovídají, co by se mělo testovat. Funkčnost je správnost chování programu. Použitelností se rozumí uživatelská přívětivost, jak snadné je s programem pracovat, zkoumá se konzistence aplikace, estetika nebo dokumentace. Spolehlivost je schopnost stejného chování programu za běžných okolností, ale také v případě výskytu chyby a větší zátěže systému. Podporovatelnost souvisí s údržbou aplikace, možností jejího případného rozšíření. V oblasti výkonu se posuzuje dostatečná rychlost dle představ uživatele. Pod ostatní vlastnosti spadá například lokalizovatelnost, kompatibilita, vymezení použitých programovacích jazyků nebo požadavky na vlastnosti fyzického vybavení počítače. [2]

3.4 Automatizované testování

Automatizací testů se rozumí vše, co nějakým způsobem usnadňuje testování a odstraňuje opakované ruční provádění stejných činností. Jedná se například o nástroje pro plánování a řízení běhu testů, nástroje pro reportování výsledků proběhlých testů nebo spouštění automatizovaných testů. Tyto programy také mohou sloužit jako podpůrná pomůcka k manuálnímu testování.

Součástí této velké skupiny nástrojů pro automatizaci jsou programy pro automatizovaný běh testů. Jde o testy, které během svého průběhu nevyžadují přítomnost testera. Typickým příkladem automatizovaných testů jsou regresní testy, kontroly zabezpečení, testy které se opakují často nebo testy, které probíhají velmi dlouhou dobu

3.4.1 Techniky automatizovaného testování

Nástroje pro automatizované testování mají různé techniky vytváření testů. Základní technikou je ruční psaní testů jazykem, který daný nástroj podporuje. Dalšími technikami, které vytváření testů usnadňují, jsou zachycení a následné přehrání aktivity uživatele, modifikace již vygenerovaných scriptů, rozpoznávání obrazu a testování řízené daty nebo klíčovými slovy.

Zachycení a přehrání

Jde o nejjednodušší a nejznámější způsob vytvoření testu, který pomocí vhodně zvoleného nástroje zvládne i uživatel bez pogramátorských dovedností. Principem takového programu je zachycení a nahrání všech činností, které uživatel v GUI provádí. Tyto záznamy lze následně přehrávat.

Modifikace scriptů

Vstupem bývají scripty automaticky vygenerované předchozí metodou, které se znalostí použitého scriptovacího jazyka můžeme dále doplňovat nebo upravovat.

Rozpoznávání obrazu

Jedná se o techniku, kde parametry jednotlivých kroků testu jsou snímky komponent, kterých se daný krok týká. Nástroje, které takové vytváření testů umožňují, rozpoznávají obraz. Takový nástroj vždy danou komponentu na obrazovce nejprve vyhledá a poté provede akci, kterou jsme určili.

Testování řízené daty

Používá se pro testy, které potřebují velké množství vstupních nebo výstupních dat. Testovací script v tomto případě načítá zvolený počet vstupních dat spolu s očekávanými výsledky a následně s nimi provede tělo testu. Typicky je možné data přijímat z textových souborů, z databází nebo z generátorů náhodných dat.

Testování řízené klíčovými slovy

Jedná se o obdobu předchozí techniky, kde navíc ve zdrojových datech nalezneme příkazy, ze kterých sestává samotný testovací script. Tyto příkazy jsou za běhu načítány a prováděny. Tím se dynamicky vytváří logika testu.

4 Nástroje pro automatizované testování GUI

Každý nástroj pro automatizované testování má řadu vlastností a funkcí. Nástroje často neslouží pouze k samotnému vytváření testů, ale podporují některé další funkce, jako je plánování testů nebo reportování jejich výsledků. 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 vytvářejí testy nebo alespoň jejich část. V popisu nalezených nástrojů se zaměřím na podporované technologie, jejich licence, podporu nástroje a způsob vytváření testů. Neméně důležitými charakteristikami je také kvalita uživatelské dokumentace či existence nějaké skupiny uživatelů, která dokáže v případě nejasností v dokumentaci pomoci.

V následujících podkapitolách je krátký popis nástrojů na současném trhu. Dále následuje přehledná tabulka těchto nástrojů spolu s nejdůležitějšími charakteristikami.

4.1 Selenium

Nástroj sloužící primárně k testování webových aplikací. Je vyvinut v Javě a je možné ho používat na různých platformách. Skládá se z několika navzájem se doplňujících částí, kterými jsou Selenium IDE, Selenium Remote Control (dále Selenium RC), Selenium Grid a Selenium WebDriver.

Selenium IDE je plugin pro internetový prohlížeč Mozilla Firefox. Oproti tomu Selenium RC je možné použít nezávisle na jediném druhu prohlížeče. Vykonává příkazy testování pomocí Selenium serveru. V těchto nástrojích se vytvářejí testy nahráváním. Selenium WebDriver je obdoba Selenia RC. Je použitelný pro všechny prohlížeče bez nutnosti využití serveru. Selenium Grid navíc umožňuje spuštění testu na několika strojích a prohlížečích zároveň. Testy je nutné v případě posledních dvou nástrojů manuálně napsat v Javě. [3]

Jedná se open-source software. Má živou komunitu uživatelů na oficiálním

diskuzním fóru. Uživatelská dokumentace je pouze v anglickém jazyce, má formu wiki stránky na oficiálním webu produktu. Jedná se o nástroj, který je stále podporován, vycházejí pravidelné aktualizace. Poslední byla vydána v říjnu roku 2015.

4.2 TestComplete

TestComplete od společnosti SmartBear umožňuje pomocí tří nezávislých modulů testovat webové aplikace, desktopové aplikace pod Microsoft Windows a mobilní aplikace vyžívající Anroid a iOS. Všechny testy mohou být ručně napsány v některém z podporovaných scriptovacích jazyků (VBScript, JScript, DelphiScript, C++Script, C#Script, Python), nebo mohou být nahrávány. [4]

Jde o placený software s možností bezplatného vyzkoušení na 30 dní. Dokumentace je v anglickém jazyce dostupná online, popřípadě po nainstalování produktu také offline. TestComplete ma aktivní komunitu uživatelů na oficiálním diskuzním fóru. Poslední verze aplikace vyšla 3. 11. 2015.

4.3 GUI Master

Nástroj GUI Master je produktem české společnosti Globtech. Testuje webové aplikace a desktopové aplikace nezávisle na platformě. Uživatel nepíše testovací skript ručně, test se sestaví z nabízených prvků a poté spustí. GUI Master mne zaujal už z důvodu, že ho vyvíjí česká firma, ale nejspíše se jedná o již neaktuální software. Uživatelské fórum neexistuje, stejně tak jakákoli zmínka o poslední vydané verzi nástroje, jeho ceně, případně nějaké možnosti bezplatného vyzkoušení. [5]

4.4 Jubula

Jubula je plugin do vývojového prostředí Eclipse nebo samostatná aplikace od firmy BREDEX GmbH. Dokáže otestovat GUI desktopových aplikací, je multiplatformní. Pomocí Jubuly také můžeme testovat webové aplikace a mobilní aplikace pod iOS. Testovací scénáře se sestavují z jednotlivých kroků a nebo nahráváním. Není nutnost znalost kódu testované aplikace, jedná se tedy o black-box testování. [6]

Aplikace je zcela zdarma. Má rozsáhlou dokumentaci a živé diskuzní fórum. Poslední verze byla vydána 9. 2. 2016.

4.5 Ascential Test

Produkt od společnosti Zeenyx slouží k testování desktopových aplikací v systému Microsoft Windows a webů napříč prohlížeči. Testové scénáře jsou sestavovány z jednotlivých kroků v editoru. [7]

Aplikace je placená bez možnosti bezplatného vyzkoušení. Nemá dostupnou dokumentaci ani uživatelské fórum. Poslední verze je z 8. 12. 2015. Společnost také nabízí produkt pro podporu manuálního testování.

4.6 Ranorex

Tento testovací nástroj od společnosti Ranorex slouží k testování desktopových aplikací, webových aplikací napříč prohlížeči i mobilních aplikací naprogramovaných širokou škálou programovacích jazyků. Je omezen na systém Microsoft Windows. Pro psaní testů jsou využívány jazyky C# a VB.NET. Testy není nutné psát ručně, je možné je nahrávat a Ranorex je poté sám převede do kódu v daném jazyce. [8]

Aplikace je placená s možností vyzkoušení demo verze. Má velmi aktivní komunitu uživatelů, přehledné webové stránky a podrobnou dokumentaci. Aktuální verze byla vydána 18. 11. 2015.

4.7 Watir

Watir je nástroj pro testování webových aplikací. Pro vytváření testů se používá scriptovací jazyk Ruby. Testovací nástroj není omezen konkrétním webovým prohlížečem, operačním systémem, ani technologií, kterou byla testovaná webová stránka vytvořena. [9]

Watir je open-source nástroj. Má dostupnou dokumentaci a aktivní diskuzní fórum. Poslední oficiální aktualizace proběhla 28. 4. 2013.

4.8 HP QuickTest

Testovací nástroj od společnosti Hewlett-Packard slouží k testování desktopových, mobilních a webových aplikací. Používá VBScript k sestavení testů, nebo nabízí možnost sestavení testovacího scénáře z jednotlivých kroků. [10]

Nástroj je placený s dostupnou zkušební verzí na 30 dní. Webová prezentace produktu je velmi nepřehledná. Je zde vcelku aktivní komunita uživatelů, kteří poradí, ale není k dohledání uživatelská dokumentace. Aktuální verze byla vydána v červnu 2015.

4.9 iMacros

Nástroj pro testování webových aplikací od společnosti iOpus má více než 10letou tradici. Může mít podobu samostatné aplikace nebo pluginu v internetovém prohlížeči. Testy jsou nahrávány jako makra, která lze následně pomocí scriptovacích jazyků dále upravovat, nebo mohou být rovnou ručně napsány. [11]

Jde o placený nástroj s 30denní zkušební verzí. Má dostupnou dokumentaci a velmi živou komunitu uživatelů. Aktuální verze vyšla 8. 10. 2015.

4.10 Maveryx

Nástroj od stejnojmenné společnosti se specializuje na desktopové aplikace naprogramované v Javě, nezávisle na platformě, a na mobilní aplikace pro systém Anroid. Může mít podobu pluginu do vývojového prostředí. Jako scriptovací jazyk používá Javu. [12]

Jde o bezplatný nástroj s možností zakoupení profesionální verze. Maveryx má velmi přehledné webové stránky, uživatelskou dokumentaci, ale poměrně neživé diskuzní fórum. Poslední verze je ze 7. 8. 2015.

4.11 QF - Test

Aplikace od Quality First Software je na trhu téměř 15 let. Podporuje testování desktopových Java aplikací nezávisle na operačním systému a webových aplikací napříč prohlížeči. Testy není nutné psát ručně, sestavují se z předpřipravených kroků nebo nahráváním. [13]

QF - Test je placený program. Je však ke stažení zkušební verze s omezenou funkčností. Má dokumentaci v anglickém a německém jazyce. Pokud by ani podrobná dokumentace nestačila, poradí také živá komunita uživatelů. Aktuální verze je dostupná od 4. 11. 2015.

4.12 Rational Functional Tester

Aplikace je od společnosti Rational Software, která spadá pod IBM. Umožňuje testování webových i desktopových aplikací nezávisle na druhu operačního systému. Testy se vytvářejí nahráváním akcí testera. Z těchto nahrávek se následně vygeneruje script v Javě nebo VB.NET, který lze později upravovat. Nabízí také možnost testování řízené daty. [14]

Aplikace je placená s 30denní zkušební verzí. Má dostupnou kvalitní dokumentaci. Je k dispozici oficiální fórum. Podpora produktu je velmi dobrá, aktualizace vycházejí několikrát do roka. Poslední byla vydána 16. 12. 2015.

4.13 Silk Test

Nástroj nabízí několik verzí produktu. Můžeme si vybrat podobu klasického programu, ve kterém testy jednoduše sestavíme z akcí. Dále nabízí možnost pluginu do vývojového prostředí Eclipse, kde se testy píší v Javě, nebo obdobného pluginu do prostředí Visual Studio s použitím C#. Je možné testování všech aplikací, omezením je operační systém Microsoft Windows. SilkTest nabízí také testování řízené klíčovými slovy. Nástroj umí rozpoznávat grafické objekty, díky tomu jím lze testovat také hry. Společnost mimo jiné také nabízí nástroj pro testovací management. [15]

Jde o placený nástroj se zkušební verzí na 45 dní. Je k dispozici dobře zpracovaná uživatelská dokumentace. Produkt má aktivní komunitu uživatelů. Aktuální verze je ze dne 16. 3. 2015.

4.14 Testing Anywhere

Jedná se o nástroj vyvíjený společností Automation Anywhere, který je na trhu již přes 10 let. Je využitelný pro testování všech druhů aplikací, desktopové jsou omezeny operačním systémem Microsoft Windows. K sestavení testů používá nahrávání akcí. Testy je možné dále upravit, nebo převést do spustitelného EXE formátu pro lepší distribuci vytvořeného testu. [16]

Produkt je placený se zkušební verzí na 14 dní. Dokumentace není dostupná online, je pouze ke stažení. Produkt má také své oficiální uživatelské fórum, ale již přes rok se zde neobjevil nový příspěvek. Poslední verze byla vydána 17. 4. 2015. Součástí produktu jsou také nástroje pro testovací management.

4.15 SikuliX

Tento nástroj nabízí možnost automatického testování aplikací nezávisle na použitém programovacím jazyku nebo platformě. K vytváření testů se využívá rozpoznávání obrazu. SikuliX díky snímkům jednotlivých komponent identifikuje objekt a může s ním pracovat. Pro psaní testů využívá jazyky Ruby a Python. Je k dispozici také jako plugin do vývojového prostředí Eclipse. [17]

SikuliX je k dispozici zdarma. Má živé diskuzní fórum a kvalitní dokumentaci. Aktuální verze vyšla 6. 10. 2015.

4.16 Tabulka nástrojů

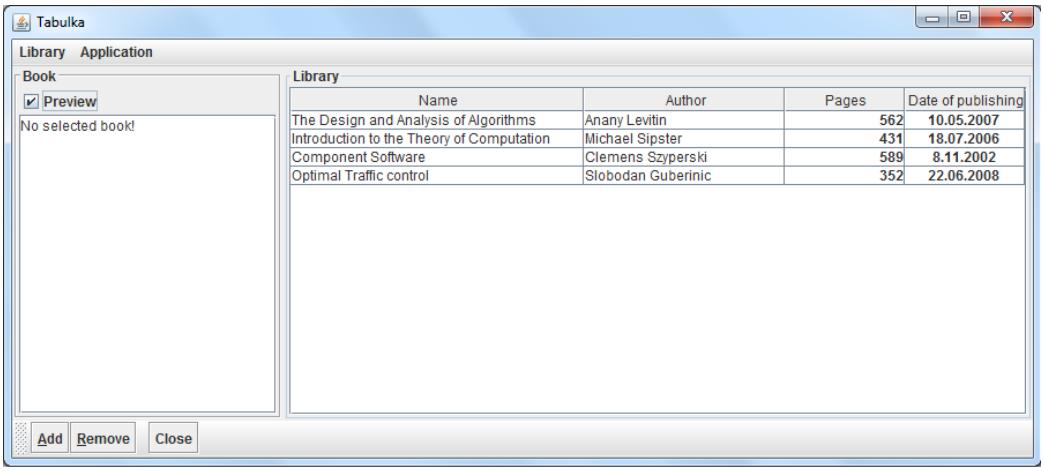
Název	**Licence**	**Technologie**	**Dokumentace**	**Podpora**
TestComplete	Trial	HTML5, Flash, .NET, Flex, Java, Siverlight, VCL, C++	✓	✓
GUI Master	Placené	Bez omezení	×	×
Ascential Test	Placené	Java, .NET, DevExpress	×	✓
Watir	Zdarma	Bez omezení	✓	×
HP QuickTest	Trial	Java, .NET, Silverlight	×	✓
Maveryx	Zdarma	Java	✓	✓
Rational Functional Tester	Trial	Java, HTML, AJAX, Flex, Silverlight, .NET	✓	×
Jubula	Zdarma	Java, HTML	✓	✓

Název	**Licence**	**Technologie**	**Dokumentace**	**Podpora**
QF - Test	Demo	Java, HTML, AJAX, jQuery	✓	✓
Selenium	Zdarma	Java, C#, Groovy, Perl, PHP, Python, Ruby	✓	✓
SikuliX	Zdarma	Bez omezení	✓	✓
iMacros	Trial	Java, JavaScript, HTML, Flash, Flex, Silverlight, .NET	✓	✓
Ranorex	Trial	Java, Delphi, .NET, DevExpress, HTML5, Flash, Flex, Silverlight a další	✓	✓
Silk Test	Trial	Java, Silverlight, HTML, AJAX, Flex, .NET	✓	✓
Testing Anywhere	Trial	Java, JavaScript, HTML Python, Silverlight, .NET, Flex, C#, C++, VB.NET a další	✓	✓

Tabulka 1: Tabulka nástrojů.

5 Porovnání vybraných nástrojů

Vybranými testovacími nástroji otestuji aplikaci napsanou v jazyce Java s využitím knihovny Swing. Jedná se o aplikaci s jednoduchým GUI, jejímž hlavním prvkem je tabulka dat, ze které lze data číst, nebo je přidávat a mazat. Obě akce lze provést tlačítkem nebo z menu. Nejprve otestuji fukční aplikaci a poté budu provádět stejné testy na téže aplikaci s úmyslně vytvořenými chybami. Chyby jsou zpravidla vytvořeny jen zakomentováním části kódu. Pro porovnání jsem vybrala nástroje SikuliX a Jubulu. Záměrně jsem vy-



Obrázek 1: Testovaná aplikace

brala právě tyto dva nástroje, protože jsou jako jediné dostupné bez omezení zcela zdarma. Na základě tohoto kritéria by se také nabízelo k porovnání Selenium. Bohužel Selenium umožňuje testování pouze webových aplikací. Porovnání s nástroji, kterými budu testovat GUI aplikace napsané v Javě, by tedy postrádalo smysl.

Všechny návody a testy jsou vytvořeny pro systém Windows 7 s nainstalovanou Javou, verze `1.8.0_73`.

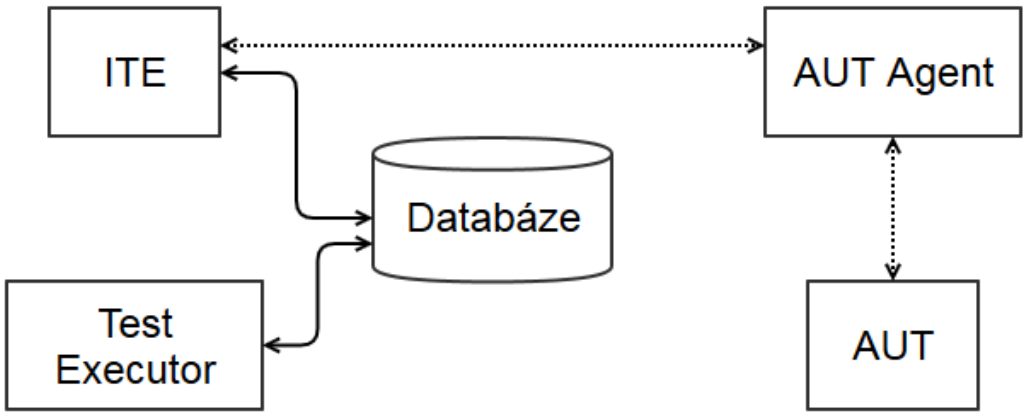
5.1 Jubula

Jubula je nástroj pro testování GUI aplikací, které jsou napsány v Javě s využitím Swingu, SWT/RCP/GEF, JavaFX, nebo v jazyce HTML. Umí také otestovat GUI aplikací iOS. Je zdarma ke stažení na svých domovských in-

ternetových stránkách [18]. Tento nástroj má velmi kvalitní podporu. Má poměrně rozsáhlou dokumentaci a nové verze aplikace pravidelně vycházejí přibližně v rozmezí šesti měsíců. Můžeme si vybrat, zda ji chceme jako samostatnou aplikaci, nebo jako plugin do vývojového prostředí Eclipse, který je produktem stejné společnosti. Pro potřeby této bakalářské práce využiji Jubulu v podobě samostatné aplikace.

5.1.1 Architektura aplikace

Jubula funguje na principu klient-server. Klientem je vývojové prostředí, kde je test napsán, kde se spouští a analyzuje jeho výsledek, tedy samotné prostředí Jubuly, ve kterém během vytváření testů pracujeme. Tento klient je označován zkratkou ITE (Integrated Testing Enviroment). Serverem je malý program nazývaný AUT Agent (Application Under Test Agent), který spouští a obsluhuje testovanou aplikaci, která se označuje zkratkou AUT (Application Under Test). AUT Agent také zabezpečuje provádění testů. Jednotlivé projekty jsou uchovávány v databázi, odkud je lze exportovat.



Obrázek 2: Architektura aplikace Jubula.

5.1.2 Zprovoznění

Jak již bylo zmíněno, pro získání aplikace je třeba navštívit oficiální internetové stránky, na kterých se pro její stažení musíme zaregistrovat. Zde si stáhneme verzi Jubuly podle platformy, na které ji chceme provozovat, a také si musíte zvolit, zda chceme samostatnou aplikaci nebo plugin do prostředí

Eclipse. Po stažení spustíme instalační balíček a řídíme se pokyny instalátoru.

Po instalaci již můžeme Jubulu spustit. Během prvního spouštění aplikace je potřeba nastavit pracovní adresář, kam se budou ukládat data. Toto umístění lze samozřejmě kdykoli později změnit podle potřeby.

5.1.3 Vytváření projektu

Abychom mohli začít vytvářet testy, musíme si nejprve vytvořit nový projekt. Překvapivě se do průvodce vytváření nového projektu nedostaneme z hlavního menu kliknutím na File, ale je potřeba v menu rozbalit nabídku Test a kliknout na položku New... Poté se objeví okno, kam začneme vyplňovat vlastnosti projektu. V prvním kroku je potřeba zvolit název projektu, zaškrnout, zda chceme v budoucnu vytvořené testovací případy uložit do knihovny a využít je také v dalších projektech. Dále máme možnost vybrat jazyk vstupních dat podle toho, jaký jazyk testovaná aplikace podporuje a v neposlední řadě si zvolíme v rolovacím menu sadu nástrojů pro testování podle technologie, kterou aplikace využívá.

V dalším kroku je potřeba vyplnit údaje o AUT, tedy o aplikaci, kterou budeme testovat. Nejprve vyplníme údaje o agentovi, který nám zprostředkuje komunikaci mezi Jubulou a testovanou aplikací. Opět se jedná o jméno, nástroj, kterým je vytvořena, jazyk a případně ID pro lepší identifikaci. Je možné tímto krokem průvodce ukončit, ale museli bychom později nastavit také cestu k AUT.

V posledním kroku tedy nastavíme cestu k `EXE` souboru nebo k souboru typu `BAT`, pojmenujeme si naší aplikaci a klikneme na Finish. U všech kroků je k dispozici srozumitelná nápověda.

5.1.4 Vytváření testů

Pokud se nám povedlo vytvořit nový projekt, nic již nebrání ve vytváření samotných testovacích případů. K tomu můžeme využít dva způsoby. Jedním je sestavení testu pomocí akcí, které si vybíráme z rozáhlého menu a pouze doplníme parametry. Druhým způsobem je náhrávání naší činnosti nad testovanou aplikací. Nedá se jednoznačně určit, který způsob je lepší. Nahrávání našich akcí je samozřejmě pohodlnější, ale nelze jím otestovat vše. Například jednoduchý test, zda je vůbec okno aplikace viditelné, těžko nahrajeme. Postupně si popíšeme, jak vytvořit test oběma způsoby. Nejprve se zaměříme na sestavování testu z předem nabízených akcí.

Výběr akcí

Po vytvoření projektu se Jubula automaticky přepne do perspektivy nazvané Functional Test Specification. Jak již název napovídá, jedná se rozhraní pro vytváření testů. V tomto rozložení okna můžme vidět v levém dolním rohu kolonku Test Case Browser. Pokud klikneme pravým tlačítkem na Test Cases, dostaneme se k možnosti New/New Test Case, tedy k možnosti vytvoření nového testovacího případu. Ten tedy vytvoříme a vhodně pojmenujeme. Zatím ale náš test nic neumí. Funkčnost mu dodáme tím, že na něj poklepeme myší a otevře se nám uprostřed okna plátno, kam můžeme dle uvážení skládat akce, které má vykonávat. Výběr těchto akcí se nachází v Test Case Browser, pokud rozbalíme strom `unbound_modules_concrete_[8.2]`.

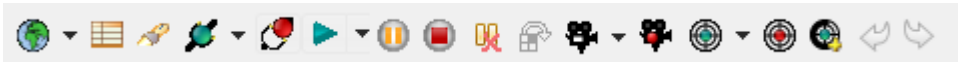
Zkusme si otestovat výběr řádky v tabulce a její následné smazání z menu Library/Remove. Nejprve vybereme řádek tabulky. K této možnosti se v Test Case Browser proklikáme pomocí Actions(Basic)->Select->Table a zde již vidíme akci, která se nazývá `ub_tbl_selectCell`. Tu jednoduše přetáhneme na plátno, kde máme otevřený náš nový testovací případ a nastavíme jí parametry. V tomto konkrétním případě se jedná o číslo řádku, sloupečku, pozice na souřadnicích X a Y, to vše spolu s jednotkami. Dále musíme vyplnit, jakým tlačítkem myši chceme na vybranou kolonku kliknout a kolikrát.

Druhou akci, kterou potřebujeme, je výběr Remove z menu. Akci najdeme v Actions(Basic)->Select->Menu Bar pod názvem `ub_mbr_selectEntry_by Textpath`. Opět přetáhneme na plátno a nastavíme parametry `TEXTPATH` a `OPERATOR` na Library/Remove a Simple match. Nesmíme samozřejmě zapomenout zkontrolovat, zda se řádek opravdu smazal. Proto opět vyberem stejný řádek, který jsme mazali a zkontrolujeme, zda se v něm nachází data z řádku, který se posunul na pozici smazaného. Kontrolu provedeme funkcí `ub_tbl_checkTextAtMousePosition`. Vše uložíme a vykonáme předposlední krok, kterým je přidání našeho nově vytvořeného testu do Test Suite Browser. To provedeme následovně. Pravým kliknutím myši do Test Suite Browser vytvoříme nový soubor testů New/New Test Suite, vhodně pojmenujeme a přetáhneme do něj náš test. Vše uložíme.

Nyní přichází chvíle, kdy je potřeba spustit AUT Agenta, který nám zprostředkuje komunikaci mezi Jubulou a aplikací, kterou budeme testovat. AUT Agent se nachází v nabídce Start ve složce s Jubulou. Zde klikneme na `Start AUT Agent`. Po jeho spuštění se nám v hlavní liště menu v Jubule zpřístupní ikona v podobě zelené tečky s úsečkou Connect to AUT Agent, na kterou klikneme pro navázání spojení s naší aplikací. Po úspěšném propojení

vidíme na témže místě modrou šipku, kterou spustíme testovanou aplikaci. Pokud je aplikace spuštěná, přejdeme k poslednímu kroku příprav a tím je mapování grafického rozhraní testované aplikace. Je nutné Jubulu seznámit s tím, jakou komponentu má testovat pomocí námi určených akcí. V hlavní liště menu, ikonou připomínající terč se zeleným středem, spustíme Start Object Mapping Mode, tím budeme přepnuti do rozhraní pro mapování.

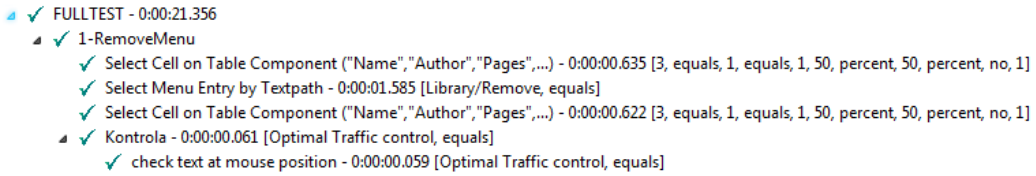
Po spuštění mapování si můžeme všimnout, že jednotlivé objekty v naší



Obrázek 3: Jubula - nabídka

aplikaci jsou zeleně orámované. To znamená, že je Jubula identifikovala. Pro namapování se nad každým zeleně orámovaným objektem zastavíme myší a stiskneme kombinaci `Ctrl+Shift+Q`. Tím se daný objekt přidá mezi namapované objekty do boxu Unassignal Technical Names. Vlevo vidíme další box nazvaný Unassigned Component Names. Zde nalezneme komponenty z předem vytvořeného testu. V našem případě se zde bude nacházet objekt tabulky a položka menu. Ty musíme ručně propojit s objekty, které Jubula identifikovala. Takto spárované objekty vidím v tabulce pod těmito boxy. Nakonec mapování ukončíme kliknutím na ikonku terče s červenou tečkou.

Po namapování můžeme přistoupit ke spuštění námi vytvořeného testu. V okénku Test Suite Browser najdeme náš test, označíme ho kliknutím na levé tlačítko myši a v téže sekci klikneme na Start Test Execution. Jubula se přepne do perspektivy nazvané Functional Test Execution a test se spustí. Je nezbytně nutné počkat, než spuštěný test proběhne a během jeho běhu s počítačem nepracovat, jinak test neproběhne správně. Po jeho proběhnutí se nám zobrazí výsledek. V tomto případě proběhl test úspěšně.



Obrázek 4: Výsledek testu smazání řádku tabulky pomocí menu.

Nahrávání akcí

Pro sestavení testu pomocí nahrávání akcí potřebujeme mít opět spuštěného a propojeného AUT Agenta s testovanou aplikací. V hlavní liště menu si tím zpřístupníme ikonku kamery Start Observation Mode, která nás přepne do režimu nahrávání. Provedeme akci, kterou chceme, aby Jubula posléze sama nasimulovala a pak zastavíme nahrávání ikonou Stop Observing Test Case. Tento nahraný test najdeme v boxu Test Case Browser. Pro spuštění testu budeme postupovat stejně jako v předchozím případě.

Jak již bylo řečeno, vše nelze otestovat pouhým nahráváním akcí. Kdybychom chtěli tímto způsobem vytvořit stejný test jako v předchozím případě, povedlo by se nám nahrát samotné smazání řádku, ale již bychom takto nedokázali otestovat hodnotu, která se po smazání v řádce nachází.

5.1.5 Spuštění již vytvořeného projektu

Pokud budete chtít vyzkoušet projekty vytvořené v rámci této bakalářské práce, je potřeba je naimportovat do Jubuly. Import projektů se provádí v menu pod nabídkou Test/Import. Po úspěšném naimportování je potřeba nastavit správnou cestu k testované aplikaci. K nastavení projektu se dostaneme opět z nabídky Test, kliknutím na položku Properties. Po otevření okna vybereme v levém menu AUTs, vybereme v okně položku Tabulka a klikneme na Edit. Opět se otevře nové okno, kde v boxu AUT configurations vybereme položku `Tabulka@localhost` a znovu klikneme na Edit. Zde si ověříme správnost cesty k souboru s koncovkou `bat` a vše uložíme.

5.1.6 Konkrétní příklad použití

Abych prozkoumala možnosti Jubuly, otestuji aplikaci zmíněnou v úvodu této kapitoly. Budu testovat mazání a přidávání řádky tabulky pomocí tlačítek a poté pomocí menu. Dále vyzkouším zobrazení okna s chybovým hlášením po zadání data vydání knížky v nesprávném formátu. Poté se ujistím, že ve třetím sloupci nelze pomocí tlačítka zadat záporný počet stránek. Nakonec se zaměřím na funkčnost checkboxu Preview, který má za úkol vypínat a zapínat podrobný náhled každého řádku.

V každém testu provedu na konci automatický restart aplikace, aby další test mohl začít opět s výchozím nastavením aplikace a pro všechny testy byly stejné podmínky. Tyto dílčí testy následně složím do jednoho souboru

testů v Test Suite Browser, který pojmenuji FULLTEST. Všechny případy otestuji na plně funkční aplikaci popsané výše a poté také na téže aplikaci s úmyslně vytvořenými chybami.

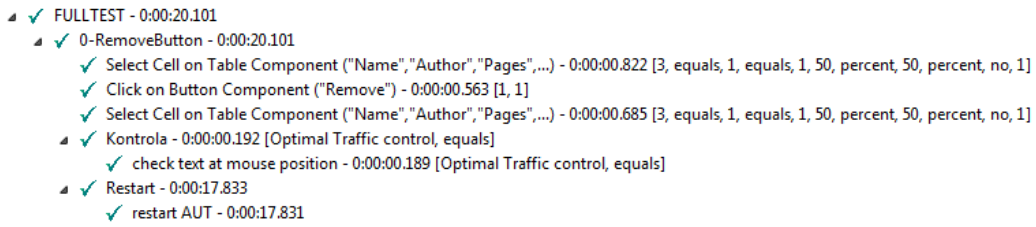
Vymazání řádku tlačítkem

Vymazání položky tabulky, ať už pomocí tlačítka `Remove` nebo z hlavního menu aplikace, budu kontrolovat následujícím způsobem. Pokud smažu třetí řádek tabulky, musí se na jeho místo posunou řádek čtvrtý. Po smazání řádku tedy budu jednoduše kontrolovat, zda je jeho obsah shodný s řádkem, který se před smazáním nacházel pod ním.

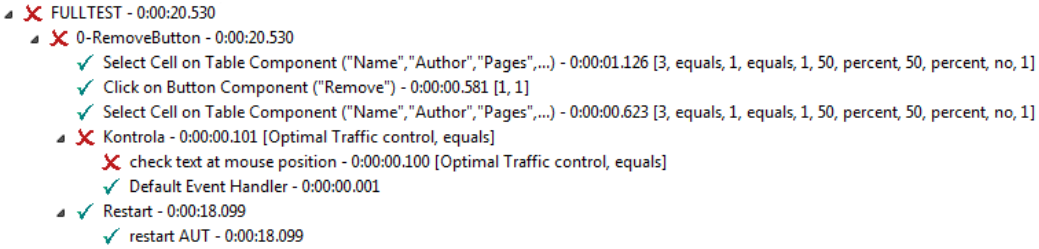
Jak vidíme ze snímku testu, který dopadl negativně, právě tato kontrola odhalila nefunkčnost tlačítka `Remove`. Samozřejmě se pro kontrolu také nabízí například práce s počtem řádků tabulky a jejich následném porovnání. Takovou jednoduchou funkci ovšem Jubula nenabízí a pokud bychom porovávali právě celkový počet řádků před akcí a po ní, nemohli bychom si ověřit, zda byl smazán konktétně námi požadovaný řádek.

Prováděné akce:

- Vybere řádek tabulky ke smazání, konkrétně řádek číslo tři.
- Klikne na tlačítko `Remove`.
- Vybere třetí řádek tabulky ke kontrole.
- Porovná obsah buňky. Obsah třetího řádku se nyní musí rovnat obsahu čtvrtého řádku, který se na jeho pozici posunul.
- Restart aplikace



Obrázek 5: Pozitivní výsledek testu smazání řádky pomocí tlačítka.



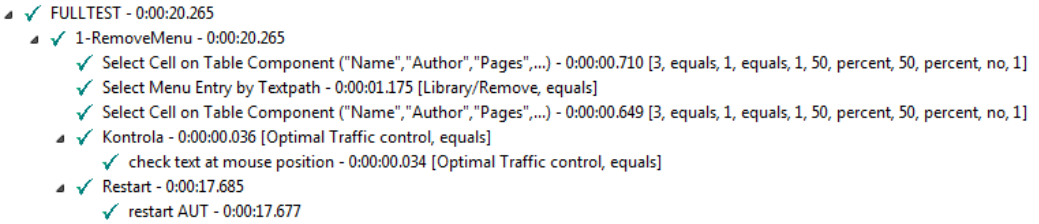
Obrázek 6: Negativní výsledek testu smazání řádky pomocí tlačítka.

Vymazání řádku pomocí menu

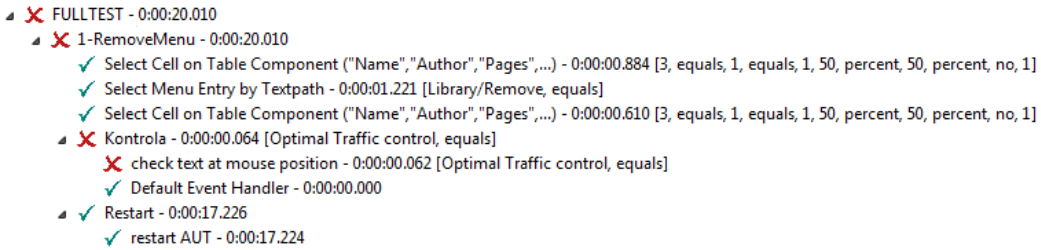
Budu postupovat obdobným způsobem jako u mazání pomocí tlačítka. Jediným rozdílem je druhý krok testu, kde místo kliknutí na tlačítko vyberu danou položku v hlavním menu aplikace.

Prováděné akce:

- Vybere řádek tabulky ke smazání, opět řádek číslo tři.
- Vybere a klikne na položku menu dle názvu `Library/Remove`, položka bude smazána.
- Vybere třetí řádek tabulky ke kontrole.
- Porovná obsah buňky. Obsah třetího řádku se nyní musí rovnat obsahu čtvrtého řádku, který se na jeho pozici posunul.
- Restart aplikace



Obrázek 7: Pozitivní výsledek testu smazání řádky pomocí menu.



Obrázek 8: Negativní výsledek testu smazání řádky pomocí menu.

Přidání řádku tlačítkem

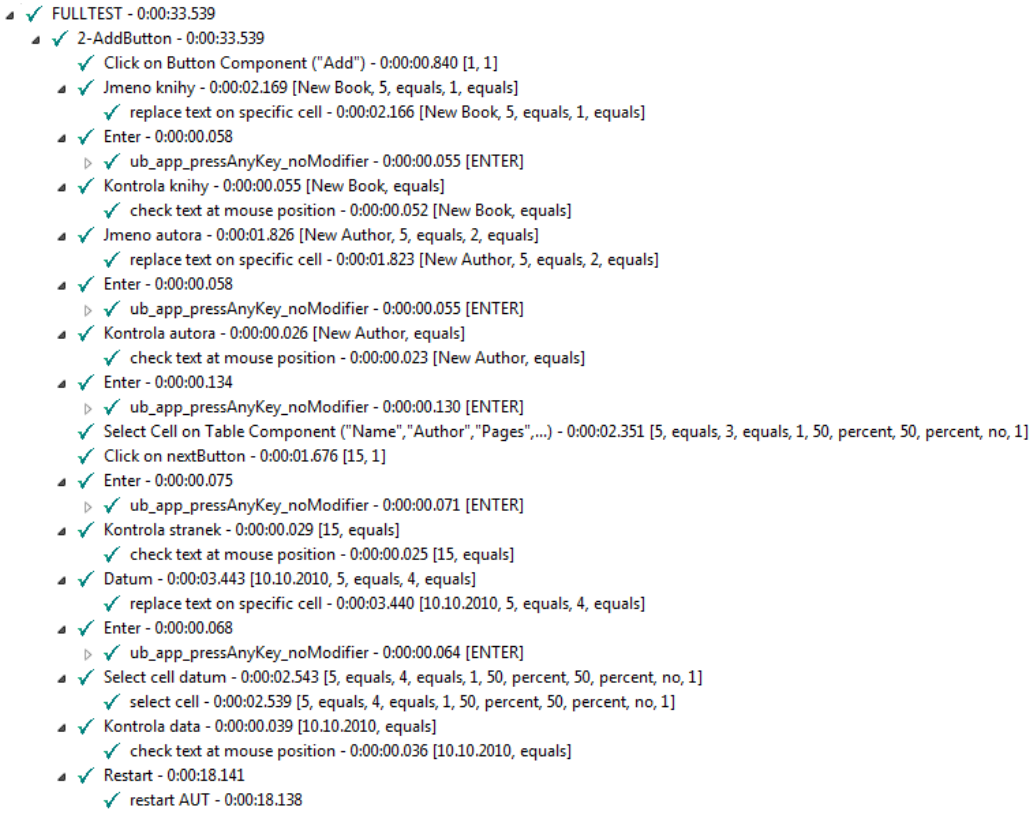
Pro kontrolu přidávání nových řádek do tabulky jsem zvolila podobný způsob jako během testování mazání položek. Po přidání nové řádky tabulky, si vyberu její jednotlivé buňky, které vyplním a poté zkontroluj, zda obsahují zadaný text.

Na snímku negativně vyhodnoceného testu vidíme, že neproběhly všechny jeho kroky. Celý test skončil v bodě, kde probíhá kliknutí na nově přidanou řádku, protože tato řádka z důvodu nefunkčnosti tlačítka `Add` pochopitelně vůbec neexistuje.

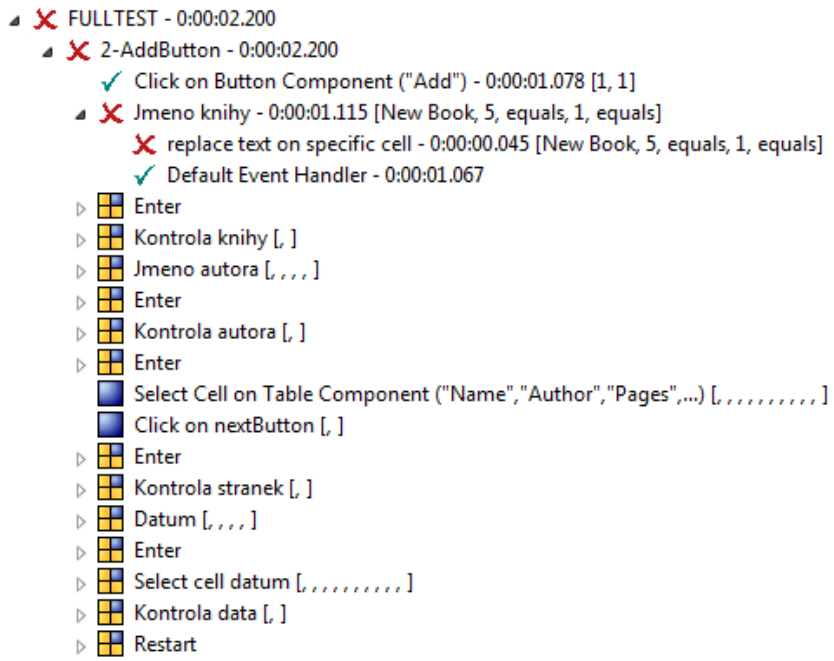
Prováděné akce:

- Klikne na tlačítko `Add`.
- Nahradí text v sloupci `Name` nově přidaného řádku.
- Stiskne `ENTER`.
- Zkontroluje, zda se obsah daného sloupce rovná námi zadané hodnotě.
- Nahradí text v sloupci `Author` nově přidaného řádku.
- Stiskne `ENTER`.
- Zkontroluje, zda se obsah daného sloupce rovná námi zadané hodnotě.
- Vybere buňku `Pages` nově přidaného řádku.
- Zadá počet stránek.
- Stiskne `ENTER`.
- Zkontroluje, zda se obsah daného sloupce rovná námi zadané hodnotě.

- Nahradí text v sloupci `Date of publishing` nově přidaného řádku.
- Stiskne `ENTER`.
- Zkontroluje, zda se obsah daného sloupce rovná námi zadané hodnotě.
- Restart aplikace



Obrázek 9: Pozitivní výsledek testu přidání řádky pomocí tlačítka.



Obrázek 10: Negativní výsledek testu přidání řádky pomocí tlačítka.

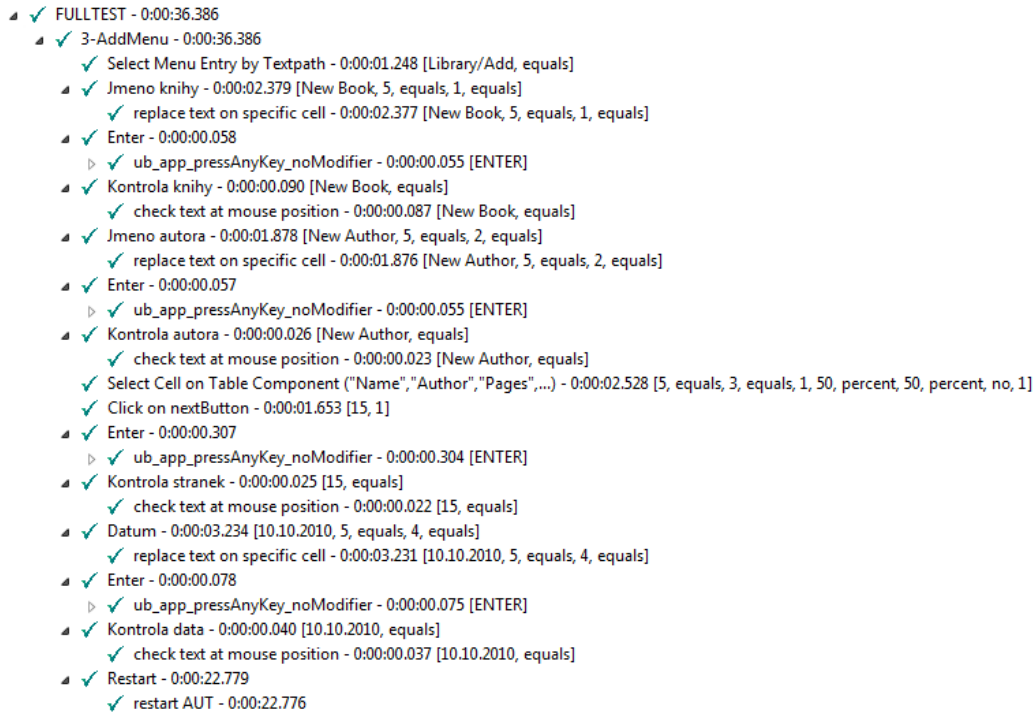
Přidání řádku pomocí menu

Budu postupovat obdobným způsobem jako u přidávání položky pomocí tlačítka. Jediným rozdílem je první krok testu, kde místo kliknutí na tlačítko vyberu danou položku v hlavním menu aplikace.

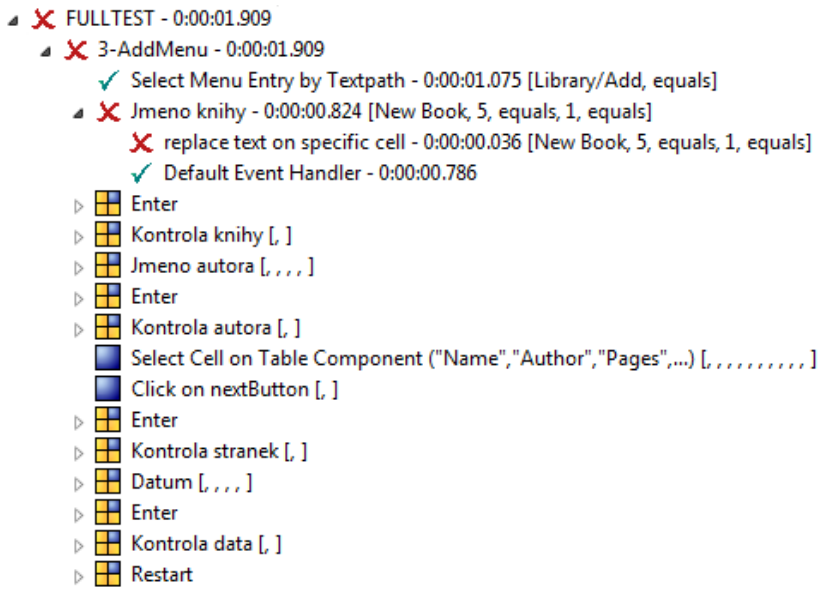
Prováděné akce:

- Vybere a klikne na položku menu dle názvu `Library/Add`, položka bude přidána.
- Nahradí text v sloupci `Name` nově přidaného řádku.
- Stiskne `ENTER`.
- Zkontroluje, zda se obsah daného sloupce rovná námi zadané hodnotě.
- Nahradí text v sloupci `Author` nově přidaného řádku.
- Stiskne `ENTER`.
- Zkontroluje, zda se obsah daného sloupce rovná námi zadané hodnotě.

- Vybere buňku `Pages` nově přidaného řádku.
- Zadá počet stránek.
- Stiskne `ENTER`.
- Zkontroluje, zda se obsah daného sloupce rovná námi zadané hodnotě.
- Nahradí text v sloupci `Date of publishing` nově přidaného řádku.
- Stiskne `ENTER`.
- Zkontroluje, zda se obsah daného sloupce rovná námi zadané hodnotě.
- Restart aplikace



Obrázek 11: Pozitivní výsledek testu přidání řádky pomocí menu.



Obrázek 12: Negativní výsledek testu přidání řádky pomocí menu.

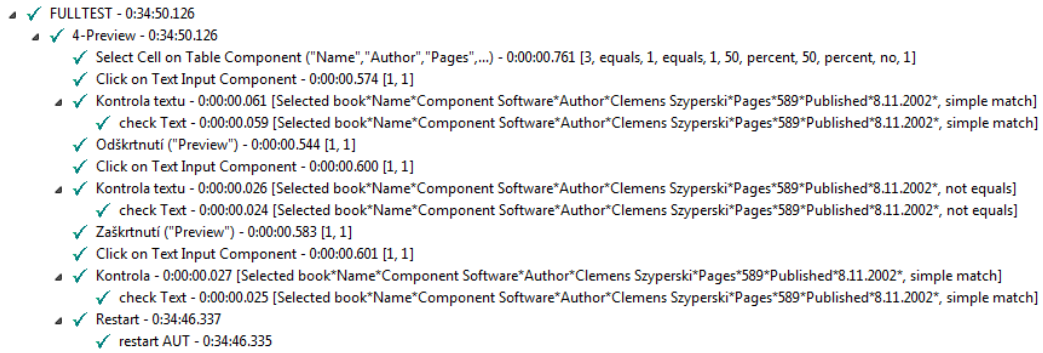
Funkčnost checkboxu

Zatrhávací tlačítko `Preview` obsluhuje textové pole, kde se zobrazuje obsah vybraného řádku. Pokud není checkbox zatržený, v poli se zobrazuje jen informace „Disabled“. Toho využiji pro kontrolu funkčnosti zatrhávacího tlačítka. Budu kontrolovat obsah textového pole se zatrženým checkboxem, kdy se v něm musí nacházet text z vybrané řádky tabulky. Po jeho odškrtnutí zkontroluji, zda obsahuje pouze řetězec, označující vypnutí této funkce, „Disabled“.

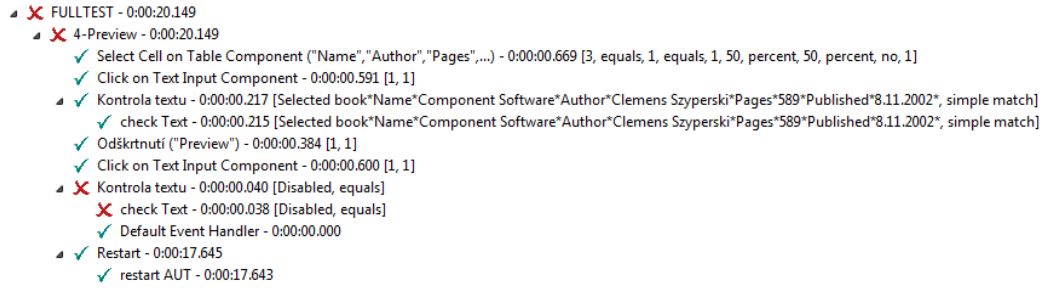
Prováděné akce:

- Vybere řádku tabulky, kterou chceme detailně zobrazit v boxu `Preview`.
- Klikne do textového pole boxu `Preview`.
- Zkontroluje, zda se obsah boxu rovná textu z vybraného řádku.
- Odškrtne checkbox `Preview`.
- Klikne do textového pole boxu `Preview`.
- Zkontroluje, zda box obsahuje jen řetězec „Disabled“.
- Zaškrtne checkbox `Preview`.

- Klikne do textového pole boxu `Preview`.
- Zkontroluje, zda se obsah boxu opět rovná textu z vybraného řádku.
- Restart aplikace



Obrázek 13: Pozitivní výsledek testu funkčnosti checkboxu.



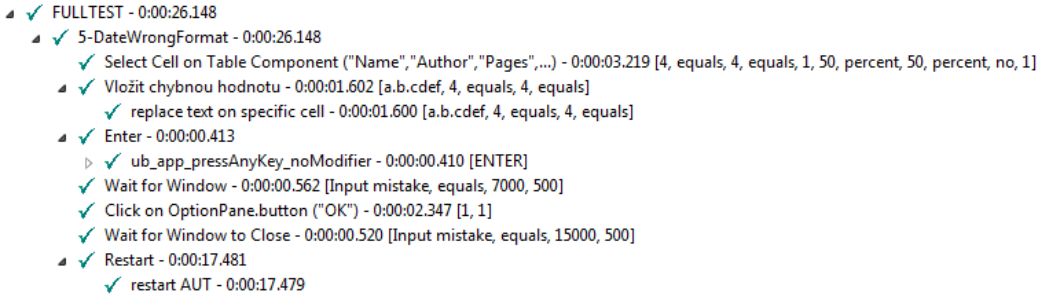
Obrázek 14: Negativní výsledek testu funkčnosti checkboxu.

Kontrola chybové hlášky

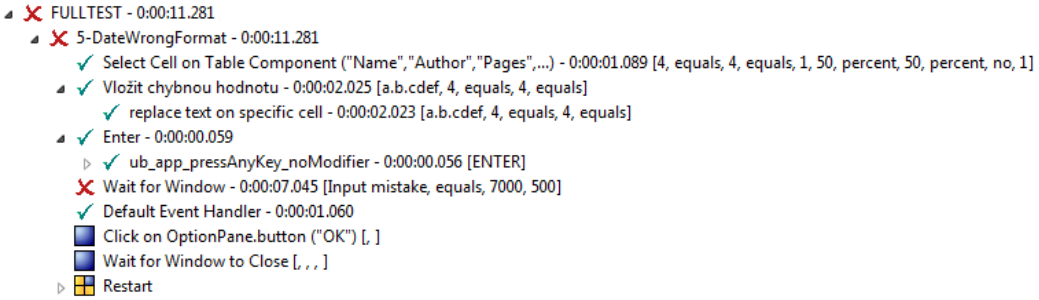
Testovaná aplikace po vložení chybné hodnoty do sloupce `Date of publishing` zobrazí chybové hlášení v podobě nového okna. Jeho přečtení je potřeba potvrdit. Budeme testovat, zda se podle očekávání toto okno otevře. V případě negativního výsledku testu se opět setkáváme se situací, že se některé akce neprovedly. Důvodem je, že se v rozbité aplikaci okno s chybovým hlášením nezobrazí. Jubula tedy nemůže kliknout na tlačítko `Ok` a počkat na zavření neexistujícího okna.

Prováděné akce:

- Vybere buňku ve sloupci `Date of publishing`.
- Vloží hodnotu, která nemá číselný tvar DD.MM.YYYY.
- Počká 7000 ms na okno, které informuje o špatném formátu vloženého textu.
- V novém okně klikne na tlačítko `Ok`.
- Počká 15000 ms na zavření okna.
- Restart aplikace



Obrázek 15: Pozitivní výsledek testu funkčnosti vyskakovacího okna.



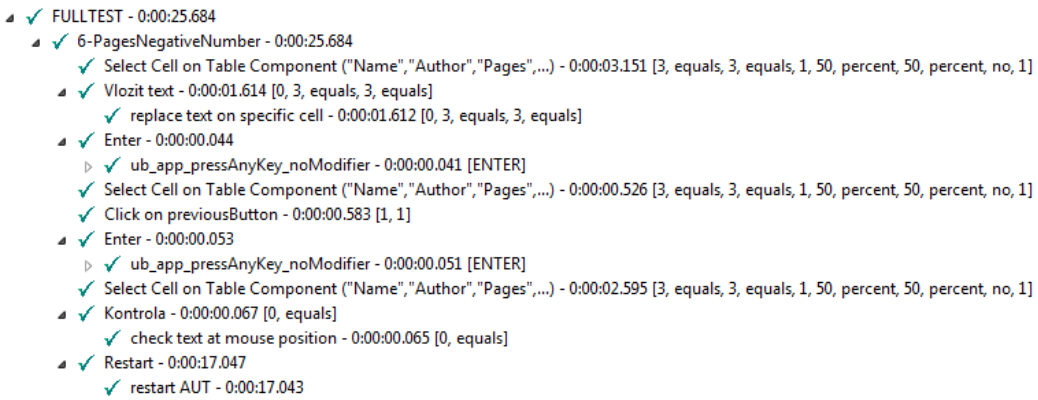
Obrázek 16: Negativní výsledek testu funkčnosti vyskakovacího okna.

Záporné hodnoty stránek

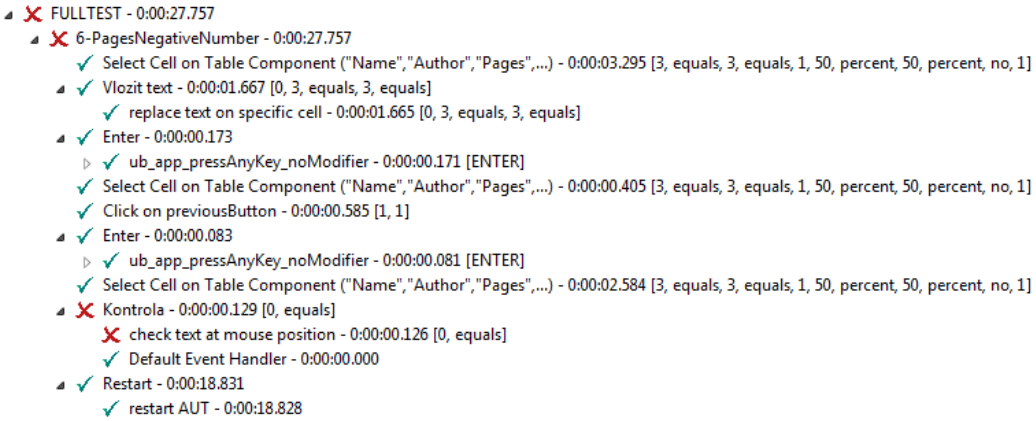
Sloupec `Pages` by měl obsahovat pouze nezáporné hodnoty. Otestuji to tak, že vyberu buňku z tohoto sloupce. Vložím do ní nulu a poté pomocí tlačítka pro ubírání stránek zkusím vložit zápornou hodnotu. Nakonec zkontroluji, že buňka stále obsahuje nulu.

Prováděné akce:

- Vybere buňku ve sloupci `Pages`.
- Vloží hodnotu `0`.
- Stiskne `ENTER`.
- Vybere stejnou buňku ve sloupci `Pages`.
- Stiskne tlačítko `previousButton`.
- Stiskne `ENTER`.
- Vybere stejnou buňku ve sloupci `Pages`.
- Zkontroluje, zda se obsah rovná nule.
- Restart aplikace



Obrázek 17: Pozitivní výsledek testu vložení záporného počtu stránek.



Obrázek 18: Negativní výsledek testu vložení záporného počtu stránek.

5.2 SikuliX

SikuliX je nástroj pro testování GUI nezávisle na technologii, kterou byla vytvořena. Dokáže otestovat vše, co vidíme běžet na obrazovce počítače. Rozeznává jednotlivé grafické objekty a dokáže s nimi pracovat bez nutnosti znalosti zdrojového kódu. SikuliX na testované obrazovce hledá dané objekty a následně nad nimi provádí námi naprogramované akce. Přesná poloha jednotlivých objektů není podstatná, objekt lze na testované obrazovce nalézt kdekoli, pokud je viditelný.

Pro vytváření jednotlivých testů a pro definování akcí nad jednotlivými komponentami lze použít jazyk Python nebo Ruby. SikuliX lze také nainstalovat jako plugin do některého z vývojových prostředí Javy, například Eclipse nebo NetBeans, pro psaní testů v Javě. Pracuje v systému Windows i Linux. Je zdarma ke stažení na svých oficiálních internetových stránkách. Tento nástroj má velmi kvalitní podporu. Má podrobnou dokumentaci a živou komunitu uživatelů, kteří se snaží poradit. S ohledem na to, jak SikuliX pracuje a nemá prakticky žádná omezení, má široký rozsah použití. Nástroj lze kromě automatického testování využít také k prosté automatizaci jakékoli aplikace s GUI. Můžeme jím ovládat například hry, nebo provádět údržbu systémů.

5.2.1 Zprovoznění

Pro stažení SikuliX se není potřeba nikde registrovat. Stačí jen navštívit oficiální internetové stránky [19] a zahájit stahování. Nejnovější verze aplikace je 1.1.0 a byla vydána 6. 10. 2015. SikuliX je Java aplikace, pro jeho provoz

je potřeba mít nainstalovanou Javu, nejlépe nejnovější verzi. Po spuštění staženého instalačního souboru je nejprve potřeba v průvodci zvolit, jakou možnost instalace požadujete. Jak již bylo zmíněno, máme na výběr samostatnou aplikaci nebo plugin do některého z vývojových prostředí. Pokud zvolíme samostatnu aplikaci, je potřeba také vybrat, zda chceme použít pro psaní testů jazyk Python nebo Ruby. Poté se SikuliX nainstaluje. Pro účely této práce využiji tohoto nástroje v podobě samostatné aplikace a použiji jazyk Python.

Po instalaci jej můžeme spustit. Nejsou potřeba žádná další nastavení, vše je připraveno k okamžitému použití. Narozdíl od Jubuly zde nejsou žádné projekty. Každý test má podobu jedné složky, která obsahuje všechny snímky testované aplikace, se kterými pracujeme a jeden zdrojový soubor, který obsahuje všechny naše příkazy a akce, které naprogramujeme. Právě tento soubor máme během vytváření testů jako pracovní.

Pokud budete chtít spustit testy, které byly vytvořeny v rámci této práce, je potřeba kliknout v menu na File/Open..., kde si poté vybereme k otevření složku s vytořeným testem. Dále je potřeba testovanou aplikaci vložit do složky `C:\\Program Files(x86)`. Pokud potřebujete testovanou aplikaci v systému z nějakého důvodu umístit jinam, je potřeba tuto cestu upravit ve zdrojovém souboru ve funkci `Restart()` a poté také v hlavní část kódu, kde se aplikace poprvé na začátku testu spustí. Doporučuji ovšem dodržet původní nastavení.

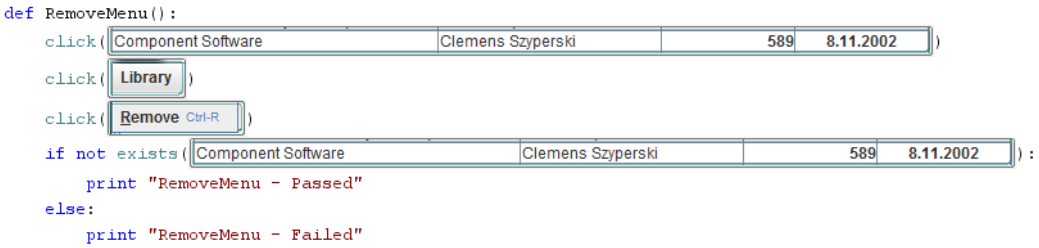
5.2.2 Vytváření testů

Po prvním spuštění SikuliX se nám automaticky vytvoří nový test, který stačí jen vhodně pojmenovat, uložit a můžeme začít s vytvářením. V levém menu vidíme nabídku akcí, které můžeme použít na snímek obrazovky. Pokud máme v tomto sloupci zatrženou možnost `Auto Capture`, automatické snímání, budeme do snímacího režimu automaticky přepnuti ihned po zvolení nějaké akce. Jak ale vidíme, jen pomocí této nabídky nějaký smysluplný test nesestavíme. Další funkčnost je potřeba ručně doprogramovat námi vybraným jazykem.

Vytvořme si nyní jeden vzorový příklad testu. Použijeme stejnou akci jako v případě Jubuly, výběr řádku tabulky a jeho následné smazání z menu. Pro výběr řádku z tabulky je potřeba na něj kliknout myší. Z nabídky si tedy vybereme akci `click()`a vytvoříme snímek řádku, který chceme odstanit. Poté přidáme ještě dvě akce stejného typu, nejprve jednu pro otevření nabídky Library a druhou pro kliknutí na položku Remove. Tato tři kliknutí stačí na

vykonání akce smazání z menu. Nyní ale přichází neméně důležitá část testu. Musíme zkontrolovat, zda se vymazání opravdu vykonalo tak, jak jsme očekávali. K tomu nám bude stačit jednoduchá podmínka, ve které se ujistíme, že smazaný řádek neexistuje. V tom případě test prošel. V opačném případě se vyskytla v testované aplikaci chyba a naším testem neprošla. Jak vidíme níže, informace po skončení testu nám příliš neřeknou, proto je přidávání vlastních výpisů opravdu žádoucí.

```
[log] CLICK on L(766,331)@S(0)[0,0 1366x768]
[log] CLICK on L(234,241)@S(0)[0,0 1366x768]
[log] CLICK on L(258,286)@S(0)[0,0 1366x768]
RemoveMenu - Passed
```



Obrázek 19: Ukázka testu v SikuliX.

5.2.3 Konkrétní příklad použití

Abych prozkoumala možnosti SikuliX, budu jím testovat stejnou aplikaci jako pomocí Jubuly. Nejprve budu testovat plně funkční aplikaci a poté její verzi s úmyslně vytvořenými chybami. Pro správné vyhodnocení použiji také stejné testovací scénáře. Stejně tak mezi jednotlivými scénáři provedu pro bezproblémový běh restart testované aplikace. Restartování zajistím naprogramováním následující funkce, kterou zavolám po každém dílčím testu.

```
def Restart():
    click(  )
    subprocess.Popen(['java', '-jar', 'c:\\Program Files\\Tabulka.jar'])
    wait(5)
```

Obrázek 20: Funkce zajišťující restartování testované aplikace.

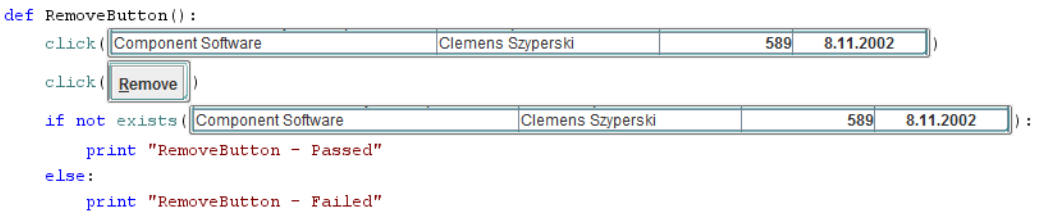
Protože vyhodnocení testů v SikuliX není bohužel ničím zajímavé a narozdíl od Jubuly nám neukáží, jaké jednotlivé kroky test obsahuje, budu u každého testu uvádět pro přehled snímek kódu testu.

Vymazání řádku tlačítkem

Vymazání položky tabulky, ať už pomocí tlačítka `Remove` nebo z hlavního menu aplikace, budu nyní testovat jinak než v případě Jubuly. Vlastnosti SikuliX nám nabízí jednodušší způsob. Stačí se jen zaměřit na mazaný řádek a po smazání zkontrolovat, zda opravdu v tabulce není.

Prováděné akce:

- Klikne na mazaný řádek.
- Klikne na tlačítko `Remove`.
- Podmínka ověřující nepřítomnost smazaného řádku.



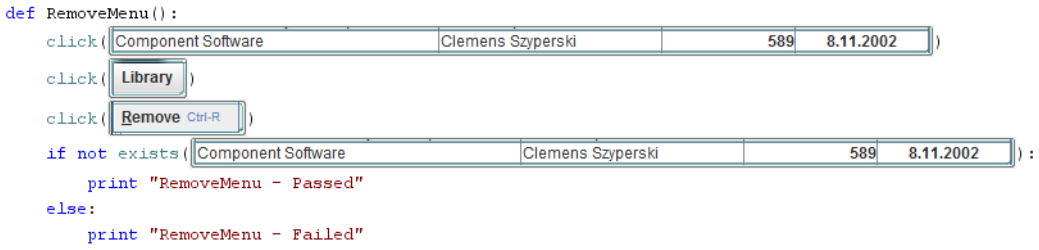
Obrázek 21: Test smazání řádku pomocí tlačítka.

Vymazání řádku pomocí menu

Tento test je shodný s naším vzorovým příkladem zmíněným výše. Vybereme řádek tabulky ke smazání. Poté v menu vybereme možnost `Library/Remove`. Nakonec zkontrolujeme, že mazaný řádek neexistuje.

Prováděné akce:

- Klikne na mazaný řádek.
- Rozbalí v menu nabídku `Library`.
- Klikne na položku `Remove`.
- Podmínka ověřující nepřítomnost odebraného řádku.



Obrázek 22: Test smazání řádku pomocí menu.

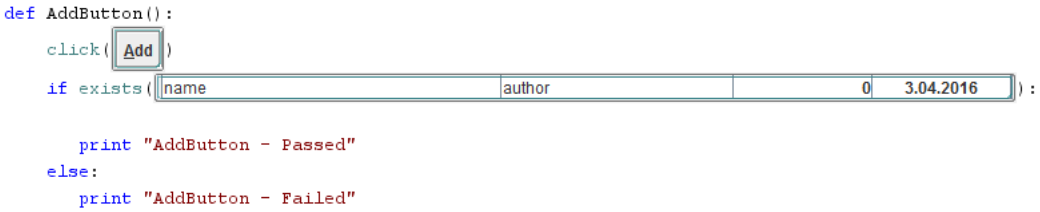
Přidání řádku tlačítkem

Otestování přidávání řádku má obdobný formát jako jeho mazání. Po kliknutí na tlačítko `Add` už se jen zaměříme na nový řádek tabulky. Zkontrolujeme, zda existuje, či nikoli. Zajímavostí je, že v případě tohoto testu můžeme vidět, jak podrobně SikuliX rozeznává snímky. Po přidání nového řádku si aplikace

sama doplní do posledního sloupce aktuální datum. To by se mohlo zdát jako problém, pokud bychom test spustili po nějakém čase znova, již by byl obsah buňky jiný a aplikace by kvůli tomu nemusela testem projít. Tak to všem není. Může to být výhodou, ale i nevýhodou. Pokud bychom chtěli pracovat s buňkou jen na základě jejího obsahu, mohlo by dojít k omylu. Proto je v těchto případech lepší pokusit se o jiné řešení, než jen využití snímků. Příklad si ukážeme dále v rámci testování možnosti zadání záporných hodnot do sloupce `Pages`.

Prováděné akce:

- Klikne na tlačítko `Add`.
- Podmínka ověřující přítomnost přidaného řádku.



Obrázek 23: Test přidání řádku pomocí tlačítka.

Přidání řádku pomocí menu

Test je postavený na stejném principu jako předchozí. Rozdíl je pouze v kliknutí do menu.

Prováděné akce:

- Rozbalí v menu nabídku `Library`.
- Klikne na položku `Add`.
- Podmínka ověřující přítomnost přidaného řádku.

```
def AddMenu():
    click( Library )
    click( Add Ctrl-A )
    if exists( name | author | 0 | 3.04.2016 ):
       print "AddMenu - Passed"
    else:
       print "AddMenu - Failed"
```

Obrázek 24: Test přidání řádku pomocí menu.

Funkčnost checkboxu

Zatrhávací tlačítko `Preview` obsluhuje textové pole, kde se zobrazuje obsah vybraného řádku. Pokud není checkbox zatržený, pole je šedivé a zobrazuje se informace „Disabled“. Pro kontrolu funkčnosti vyberu kliknutím řádek k zobrazení a zkontroluju, za textové pole zobrazuje informace. Poté kliknu na checkbox a zkontroluji, zda je textové pole neaktivní. Použiji pro to dvě vnořené podmínky.

Prováděné akce:

- Klikne na řádek, který má být podrobněji zobrazen.
- Zkontroluje zobrazení informací v textovém poli.
- Odškrtne checkbox `Preview`.
- Zkontroluje neaktivitu textového pole.

```
def Preview():
    click( Introduction to the Theory of Computation | Michael Sipster | 431 | 18.07.2006 )
    if exists( ):
        click( )
        if exists( ):
            print "Preview - Passed"
        else:
            print "Preview - Failed"
    else:
        print "Preview - Passed"
```

Obrázek 25: Ověření funkčnosti checkboxu.

Kontrola chybové hlášky

Pokusíme se vložit text do sloupce `Date of publishing`, kde aplikace očekává číselnou hodnotu ve tvaru DD.MM.YYYY. Pokud uživatel vloží nesprávný vstup, aplikace zareaguje objevením nového okna s chybovou hláškou. Pro SikuliX neexistuje nic jednoduššího, než kontrolovat existenci nějakého grafického prvku.

Prováděné akce:

- Dvakrát klikne na buňku ve sloupci `Date of publishing`.
- Vloží text „a.b.cdef“.
- Stiskne `ENTER`.
- Pokud se zobrazí okno s hláškou, potvrdí ho.

```
def Date():
    doubleClick( 18.07.2006 )
    type("a.b.cdef")
    type (Key.ENTER)
    if exists( <image> ):
        click( OK )
        print "Date format - Passed"
    else:
        print "Date format - Failed"
```

Obrázek 26: Ověření funkčnosti chybové hlášky.

Záporné hodnoty stránek

Tímto testem zjistíme, zda do sloupce `Pages` nelze vložit záporné číslo. Zde se setkáváme s problémem SikuliX. Rozhodla jsem se editovat buňku s číslem „352“. Bohužel ne každý běh tohoto testu byl správný. Stávalo se, že SikuliX vybral jinou buňku, nejčastěji tu s obsahem „562“. Proto jsem musela upřesnit lokaci, kde má SikuliX hledat pomocí příkazu `Location()`. Pak do buňky napíši nulu a pokusím se dvojitým kliknutím myši na tlačítko s šipku dolů, znázorňující ubírání, vložit záporné číslo. Poté vyberu obsah buňky klávesovou zkratkou `CTRL+A` a zkopíruji do schránky `CTRL+C`. Obsah schránky

uložím do proměnné, kterou zkontroluji, zda obsahuje stále nulu. Pokud je hodnota nulová, testovaná aplikace testem prošla.

Prováděné akce:

- Výběr lokace hledání.
- Dvojitý klik do buňky.
- Napíše nulu do buňky.
- Dvakrát stiskne tlačítko pro ubrání počtu stránek.
- Zkopíruje obsah buňky do schránky.
- Uloží obsah schránky do proměnné.
- Porovná obsah proměnné.

```
def Pages():
    Location(930, 349)
    doubleClick(                352)
    type("0",KeyModifier.SHIFT)
    doubleClick(    )
    type("a",KeyModifier.CTRL)
    type("c",KeyModifier.CTRL)
    page = Env.getClipboard().strip()
    if page == "0":
        print "Pages - Passed"
    else:
        print "Pages - Failed"
```

Obrázek 27: Ověření vstupu ve sloupci „Pages“.

5.3 Zhodnocení

5.3.1 Jubula

Na první dojem na mne zapůsobila lépe Jubula. Nabízí velké množství akcí, ze kterých lze během sestavování testů vybírat, včetně testování vlastností objektů. Bohužel po bližším prozkoumání této konkrétní funkce se ukázalo, že

testuje pouze vlastnosti grafických komponent jako je například jejich výška, šířka a barva. Přestože Jubula částečně vidí do zdrojového kódu testované aplikace, neumožňuje otestovat například hodnotu boolové proměnné. Nemá také přístup k metodám daného objektu. Této možnosti bych určitě využila během testování funkčnosti zatrhávacího tlačítka v testované aplikaci. Bylo by to spolehlivé a vytvořený test by se dal použít i po případných úpravách aplikace, kde by se mohla změnit informační hláška v případě neaktivního boxu, který je tímto tlačítkem obsluhován.

Za velkou výhodu Jubuly považuji možnost vytváření testů pomocí nahrávání akcí. Zejména některé kroky v rámci testů, jako vybírání řádků tabulky, mi nahrávání akcí velmi usnadnilo. Právě během vybírání řádku tabulky se v případě ručního sestavování testu z nabízených akcí nastavuje velké množství parametrů a chybí zde možnost jednotlivé kroky kopírovat. Proto bylo jejich nahrání časově výhodnější a dalo se tak i vyhnout případným překlepům v nastavovaných hodnotách.

Naprostro nedostačující je kvalita dokumentace Jubuly. Zpočátku se svojí rozsáhlostí jevila jako použitelná, ale po hledání nějakého konkrétního řešení problému jsem dospěla k závěru, že dokumentace mi nepomůže. Naopak musím vyzdvihnout komunitu uživatelů, kteří působí na diskuzním fóru tohoto projektu. V případě problému se snažili poradit ze všech sil. Jediným problémem byla nutnost schválení prvního příspěvku nového uživatele administrátorem stránek. Schválení trvalo téměř dva dny. Poté již ale nic nebránilo v dotazování se.

Za zhodnocení stojí také samotné uživatelské prostředí Jubuly. Již na první pohled si nejde nevšimnout jisté podobnosti s vývojových prostředím Eclipse. Dá se to očekávat, Jubula je vyvýjena stejnou společností. Celé okno Jubuly je rozděleno do několika boxů. Chvíli trvá, než se uživatel zorientuje. Za intuitivní bych její ovládání určitě neoznačila. Nepomáhá tomu také množství kroků, nutných pro vytvoření nového projektu a velký počet akcí, které uživatel musí vykonat pro spuštění testované aplikace. Z tohoto důvodu také Jubula dopadla hůře v měření časů potřebných k vytvoření a spuštění testu.

Tento nástroj si umím představit pro testování aplikací s jednodušším GUI, jako jsou databáze nebo informační systémy. Je však nutné dodržet hlavní podmínku. Testovaná aplikace musí být vytvořena v Javě nebo pomocí HTML.

5.3.2 SikuliX

SikuliX nabízí mnohem přívětivější uživatelské rozhraní. Je jednoduché a intuitivní. Oproti Jubule obsahuje hlavní okno jen dva boxy, jedním je nabídka předpřipravených akcí a druhým je již samotné pracovní plátno. Jednoduchost se dá v tomto případě očekávat, protože akcí na výběr není mnoho a samotná funkčnost testu se musí ručně doprogramovat. Jednodušší je také spouštění a zakládání projektů, není potřeba nic dlouze nastavovat a zapínat. Staší jen zapnout SikuliX a testovanou aplikaci a vše je připraveno.

Jako nepříjemný se může jevit automatický přechod do režimu snímkování po vybrání nějaké akce z menu. Často se stane, že má uživatel kromě SikuliX a testované aplikace spuštěný ještě jiný program a nemá v okamžiku snímání obrazovky viditelnou aplikaci, kterou testuje. Jako nevýhodu vidím ikonu spuštěného SikuliX na hlavní liště. Není nijak vyřešena, vypadá jako ikona každé jiné aplikace napsané v Javě. Pokud testujete aplikaci, která je také napsána v Javě, je to velmi nepříjemné.

S dokumentací tohoto nástroje jsem neměla žádný problém. Vše, co jsem potřebovala, bylo snadno k nalezení. Vysvětlení jsou velmi podrobná a nechybějí ani příklady. I v tomto případě jsem vyzkoušela pomoc od uživatelů na diskuzním fóru. Šlo zde přispívat okamžitě, nebylo potřeba žádné prvotní schválení či ověření a komunita je zde také velmi nápomocná.

Z mého pohledu je SikuliX opravdu výkonný nástroj z důvodu možnosti doprogramování jakékoli akce. Z tohoto důvodu si umím SikuliX představit nejen jako nástroj pro automatické testování, ale také jako způsob, kterým lze automatizovat mnoho dalších často vykonávaných akcí. Například automatické instalace různých programů, vzdálená správa nějakého počítače, serveru nebo automatické vykonávání akcí v různých hrách. Obrovskou výhodou proti Jubule je také jeho naprostá nezávislot na technologiích, kterými byla testovaná aplikace vytvořena.

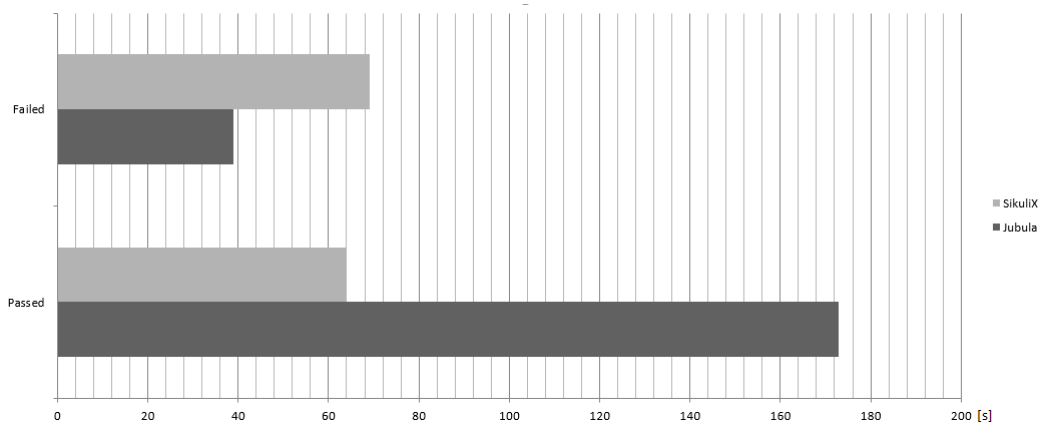
5.3.3 Časy běhu

Pro lepší představu časové náročnosti běhu testů v jednotlivých nástrojích jsem provedla následující měření. Nechala jsem desetkrát proběhnout celkový test, který obsahoval všechny vzorové testy z předchozích kapitol. Test běžel nejprve nad bezchybnou aplikací a poté nad aplikací se záměrně vytvořenými chybami. Měření probíhalo na notebooku s 64bitovým systémem Windows 7 Professional, 4 GB RAM, dvoujádrový procesor Intel Celeron B81 s taktem

1,6 GHz.

Jubula dosáhla průměrného výsledku u bezchybného testu 2 minuty 53 sekund. Testy, které skončily chybou, proběhly v nesrovnatelně kratším čase, průměrně za 39 sekund. Důvodem je, že Jubula v případě chyby v některých testech již nevykonává akce, které za případnou chybou následují. Tato skutečnost již byla zmíněna a vysvětlena v kapitole 5.1.6.

SikuliX má vyrovnané výsledky v obou případech. Bezchybné testy běží průměrně 1 minutu 4 sekundy, testy ukončené chybou pak 1 minutu 9 sekund.



Obrázek 28: Časy běhu testů v jednotlivých nástrojích.

6 Uživatelská použitelnost

Pro zhodnocení uživatelské použitelnosti dvou vybraných nástrojů jsem vybrala pět dobrovolníků. Připravila jsem návod, jak pracovat s Jubulou a se SikuliX. Součástí návodu byly dva testy, které měli dobrovolníci podle návodu sestavit. Jednalo se o otestování zobrazování okna s chybovou hláškou po zadání špatného formátu data do buňky ve sloupci „Date of publishing“. Druhým úkolem bylo otestovat přidávání řádku tabulky pomocí menu testované aplikace. Poté následovalo sestavení třetího testu bez návodu. Jednalo se o obdobu druhého úkolu. Testovalo se mazání řádku tabulky pomocí menu. Použitý návod i samostatný úkol jsou součástí přílohy. Z naměřených časů

	Jubula			**SikuliX**		
	Úkol 1	**Úkol 2**	**Úkol 3**	**Úkol 1**	**Úkol 2**	**Úkol 3**
Osoba 1	17 min	11 min	14 min	10 min	9 min	4 min
Osoba 2	9 min	6 min	12 min	7 min	4 min	6 min
Osoba 3	10 min	7 min	10 min	4 min	5 min	4 min
Osoba 4	14 min	8 min	15 min	13 min	7 min	3 min
Osoba 5	11 min	9 min	13 min	9 min	5 min	4 min
Průměr	12 min 12 s	8 min 12 s	12 min 48 s	8 min 36 s	6 min	4 min 12 s
MIN	9 min	6 min	10 min	4 min	4 min	3 min
MAX	17 min	11 min	15 min	13 min	9 min	6 min
Median	11 min	8 min	13 min	9 min	5 min	4 min
Rozptyl	8 min 34 s	2 min 58 s	2 min 58 s	9 min 24 s	3 min 12 s	58 s
Odchylka	2 min 55 s	1 min 43 s	1 min 43 s	3 min	1 min 47 s	57 s

Tabulka 2: Tabulka měření dobrovolníků.

můžeme vidět, že SikuliX dopadlo ve všech případech měření mnohem lépe než Jubula. Samozřejmě by se dalo namítat, že svůj podíl na tom má také doba běhu jednotlivých testů, jak jsme mohli vidět v předchozí kapitole. Ale je třeba vzít v úvahu, že v předchozí kapitole byly měřeny časy běhu testovacího případu, který měl celkem deset testů. Naši dobrovolníci ovšem spouštěli vždy jen jeden test. V tomto případě je časový rozdíl nepatrný. Větší vliv má spíše množství kroků, které předcházejí spuštění každého testu. Větší časovou náročnost má také na svědomí rozlehlejší grafické rozhranní Jubuly. Účastníkům měření trvalo velmi dlouho, než v Jubule našli, co potřebovali, často se museli i několikrát vracet k návodu a opakovaně si přečíst, jak postupovat.

Čtyři z pěti účastníků měření hodnotili lépe SikuliX. Důvodem bylo čisté přehledné rozhraní a oceňovali možnost doprogramování jakékoli funkce. Objevovala se i prohlášení, že SikuliX určitě využijí v budoucnu jinak, než je jeho primární účel. Jako příklad bych uvedla zajímavý nápad s automatizováním her na internetu, které se ovládají jen klikáním na levé tlačítko myši.

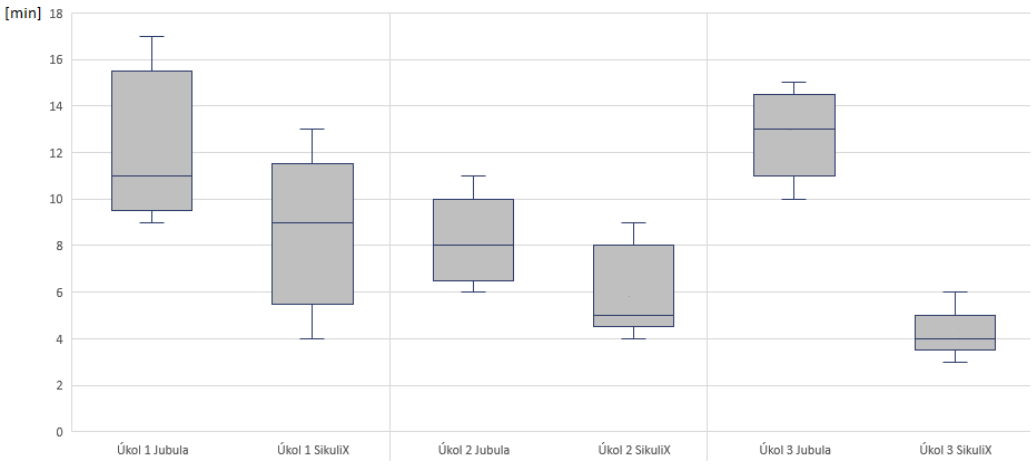
I přes velmi kladné hodnocení se objevovaly problémy. Týkaly se zejména automatického režimu snímkování. Účastníkům se často stávalo, že po přepnutí do režimu snímkování neměli na ploše viditelnou testovanou aplikaci. Problém měli všichni také s testem, který kontroluje funkčnost mazání řádku tabulky pomocí menu. Pokud si rozklikneme nabídku „Library“, abychom si zviditelnili položku „Remove“, a vrátíme se zpět do SikuliX, abychom vytvořili snímek, rozbalená nabídka zmizí. Zde tedy nestačí režim automatického snímkování, ale je potřeba položku menu „Remove“ vyfotit klasicky ručně pomocí `Print Screen` a obrázek vložit do některého z grafických editorů a snímek oříznout do požadovaného tvaru. Dalším problémem bylo špatné rozeznávání snímků, tento problém jsem již zmínila v kapitole 5.2.3. U dvou z pěti účastníků nastala situace, kdy SikuliX špatně nalezlo buňku s konkrétním datem.

V hodnocení Jubuly se shodně objevovaly výtky vůči nemožnosti kopírování kroků testu a nefunkčnímu tabulátoru v případě vyplňování tabulky parametrů. Je potřeba mezi jednotlivými řádky tabulky parametrů přepínat myší. Velkým problémem byla náročnost spouštění testů, kterému předchází start AUT agenta, propojení aplikace a až poté možnost jejího spuštění a spuštění testu.

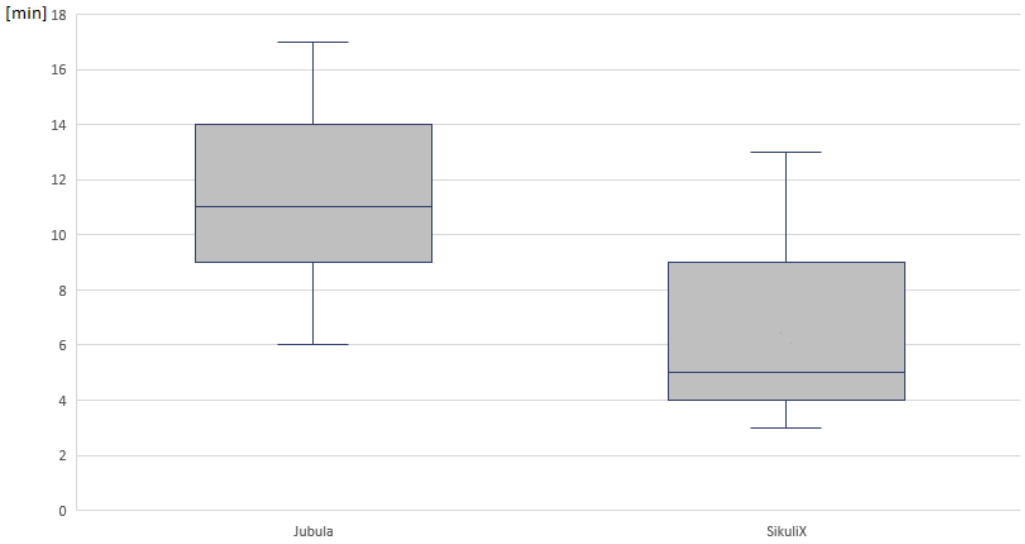
Jediný účastník měření označil Jubulu za lepší nástroj. Uvedl, že mu Jubula připadá přehledná, protože mu připomíná rozhraní nástroje Eclipse. Měl také pocit, že Jubula nabízí větší množství akcí a tím pádem je prý vhodnější pro komplexní testování aplikace. I jemu vadilo chybějící kopírování jednotlivých kroků testu. Tento člověk považoval za nedostatek SikuliX právě snímkování. Vadilo mu, že musí v testované aplikaci všechny situace, se kterými v SikuliX pracuje, sám vytvořit, aby je mohl vyfotit. Pravdou je, že to značně zpomaluje tvorbu testů. Například během testování správného zobrazení okna s chybovým hlášením bylo potřeba vložit špatný text pro vyvolání okna a až poté ho vyfotit.

Následující grafy zobrazují přehledně průměrnou časovou náročnost sestavení jednotlivých testů. První graf znázorňuje průměr času k sestavení každé úlohy v obou nástrojích. Vždy je vedle sebe ke srovnání čas úkolu nejprve v Jubule a poté v SikuliX. Obdélníková část grafu vymezuje první a třetí kvartil. Linie vycházející zespodu a shora znázorňují minimum a maximum naměřených hodnot. Úsečka v obdélníku označuje medián naměřených hodnot. V druhém grafu můžeme vidět celkový průměrný čas na vytvoření testu v každém nástroji na základě měření dobrovolníků.

Je jasně vidět, že SikuliX dopadlo ve všech případech časově lépe než Jubula. Z časových údajů o době zpracovávání třetího samostatného úkolu se dá usoudit, že dobrovolníci rychleji pochopili, jak se SikuliX pracovat, a nebylo pro ně tak náročné sestavit další test bez návodu. U Jubuly naopak vidíme, že třetí úkol trval účastníkům pokusu nejdéle. Většina lidí byla během jeho zpracovávání zmatená a dlouho jim trvalo vyhledat jednotlivé kroky testu v nabídce Jubuly.



Obrázek 29: Graf časové náročnosti pro vytvoření jednotlivých úloh.



Obrázek 30: Graf průměrné náročnosti.

6.1 Shrnutí hodnocení nástrojů

Závěrem předchozích kapitol je tvrzení, že SikuliX je lepším nástrojem pro automatické testování GUI než Jubula. Je uživatelsky přívětivější a přehlednější. Jeho použití je intuitivnější a snadno pochopitelné. Díky ručnímu programování testů má nesrovnatelně širší oblast využití. Závěrečný pokus s pěti dobrovolníky tento výsledek podpořil. V SikuliX lze stejné úlohy vytvořit v rychlejším čase s menším počtem kroků. Také následné spuštění těchto testů je snadnější a doba jejich běhu rychlejší. Velkou výhodou je nezávislost na technologii, kterou byla vytvořena testovaná aplikaci. Pomocí SikuliX se dá otestovat nebo automatizovat prakticky cokoli.

7 Závěr

Tématem práce byl „Přehled nástrojů pro automatické testování GUI“. Jejím cílem bylo vytvořit přehled nástrojů pro automatické testování GUI, vybrání několika z nich k detailnější analýze a vytvoření ukázkových příkladů. Součástí práce bylo také posouzení jejich uživatelské použitelnostii.

V teoretické části této práce jsem se věnovala vysvětlení základních pojmů v oblasti testování a jeho automatizace. Popsala jsem různé druhy uživatelských rozhraní. Dále jsem se věnovala důvodům, proč testovat GUI a jaké se v této oblasti nejčastěji objevují chyby. Posledními tématy v teoretické části práce byly metody využívané v testování a následně jeho automatizace a techniky, které se v automatizaci testování využívají.

Na dnešním trhu je velké množství nástrojů pro automatické testování aplikací s GUI. V této práci jsem provedla průzkum trhu a uvedla nejdůležitější informace o vybraných nástrojích. Tyto informace mohou pomoci při výběru vhodného produktu pro testovanou úlohu. Popsala jsem technologie, které každý nástroj podporuje, princip, na jakém funguje a zjišťovala jsem také kvalitu podpory. Poté jsem si vybrala nástroje Jubulu a SikuliX pro podrobnější analýzu. Tyto nástroje jsem si vybrala, protože jsou každému dostupné zdarma a mají stále aktivní podporu. Také využívají rozdílné techniky pro vytváření testů.

Nejprve jsem popsala, kde tyto dva nástroje bezplatně získat a způsob, jakým je zprovoznit. Následně jsem v každém nástroji vytvořila sedm testů, kterými jsem ověřovala funkčnost testované aplikace. Kroky jednotlivých testů jsem popsala podrobně a přiložila i fotografie. Díky tomu by měl bez problémů vytvořit obdobné testy i uživatel bez předchozích zkušeností s těmito nástroji. Jubulu i SikuliX jsem se snažila objektivně posoudit a popsat všechny problémy, se kterými jsem se během práce s nimi setkala. Nakonec jsem provedla měření časů běhu mnou vytvořeného souboru testů, který obsahoval již zmíněných sedm vzorových testů.

Poslední částí práce bylo zhodnocení uživatelské použitelnosti Jubuly a SikuliX. Pro tento účel jsem vybrala pět dobrovolníků, kteří se účastnili měření. Součástí měření bylo vytvoření dvou testů, ke kterým měli k dispozici podrobné návody. Posledním úkolem bylo vytvoření třetího testu, tentokrát bez napsaného postupu. Všech účastníků jsem se také zeptala na jejich názory na jednotlivé nástroje, jak se jim s nimi pracovalo a jakých nedostatků si během

práce všimli. Výstupem měření je slovní zhodnocení a přehledné grafy časové náročnosti.

Hlavním přínosem této práce je shrnutí základních teoretických poznatků v oblasti testování a přehledný seznam dostupných nástrojů na současném trhu. Dále podrobnější analýza dvou z nich a vytvoření srozumitelných příkladů použití.

Seznam obrázků

Literatura

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

[2] ROUDENSKÝ, Petr a Anna Havlíčková. *Řízení kvality softwaru: průvodce testováním.* 1. vyd. Brno: Computer Press, 2013. ISBN 978-80-251-3816-8.

[3] *Selenium - Web Browser Automation.* [online]. 2016 [cit. 2016-04-26]. Dostupné z: http://www.seleniumhq.org/

[4] Automated Software Testing. *Software Testing and Test Automation Tool.* [online]. SmartBear Software, 2016 [cit. 2016-04-26]. Dostupné z: https://smartbear.com/product/testcomplete/overview/

[5] Testovací nástroje – GTL GUI Master. *GTL GUI Master - unique test tool for applications tests GTL Tools.* [online]. Globtech [cit. 2016-04-26]. Dostupné z: http://www.gtltools.com/testovaci-nastroje/test-tools-gui-master-cs/?lang=CS

[6] *Jubula The functional testing tool.* [online]. BREDEX, 2016 [cit. 2016-04-26]. Dostupné z: http://testing.bredex.de/

[7] AscentialTest. *Software Testing Tools: Automated Testing, Manual Testing and Test Management.* [online]. Zeenyx, 2015 [cit. 2016-04-26]. Dostupné z: http://www.zeenyx.com/AscentialTest.html

[8] Ranorex. *Test Automation for GUI TestingRanorex.* [online]. Ranorex, 2016 [cit. 2016-04-26]. Dostupné z: http://www.ranorex.com/

[9] Watir.com. *Web Application Testing in Ruby.* [online]. [cit. 2016-04-26]. Dostupné z: https://watir.com/

[10] Hewlett Packard Enterprise. *Automated Testing, Unified Functional Testing, UFT* [online]. Hewlett Packard Enterprise Development,

2016 [cit. 2016-04-26]. Dostupné z: http://www8.hp.com/us/en/software-solutions/unified-functional-automated-testing/

[11] iMacros Software. *Browser Automation, Data Extraction and Web Testing.*[online]. iMacros, 2016 [cit. 2016-04-26]. Dostupné z: http://imacros.net/

[12] Maveryx. *Maveryx – Software Testing is a Life Style.* [online]. Maveryx - Test Automation Framework, 2016 [cit. 2016-04-26]. Dostupné z: http://www.maveryx.com/

[13] QFS Quality First Software. *GUI Test Automation for Java and Web with QF-Test.* [online]. QFS, 2016 [cit. 2016-04-26]. Dostupné z: https://www.qfs.de/

[14] IBM - United States. *IBM - Rational Functional Tester.* [online]. IBM, 2016 [cit. 2016-04-26]. Dostupné z: http://www-03.ibm.com/software/products/cs/functional

[15] Micro Focus. *Functional Test Automation – Silk Test.* [online]. Micro Focus, 2016 [cit. 2016-04-26]. Dostupné z: http://www.borland.com/en-GB/Products/Software-Testing/Automated-Testing/Silk-Test

[16] Automation Anywhere. *Automation Testing Tools and Software | Testing Anywhere.* [online]. Automation Anywhere, 2015 [cit. 2016-04-26]. Dostupné z: https://www.automationanywhere.com/testing/

[17] *Sikuli Script - Home.* [online]. 2016 [cit. 2016-04-26]. Dostupné z: http://www.sikuli.org/

[18] BREDEX GmbH. *Download Jubula software and documentation.* [online]. Braunschweig: BREDEX, 2016 [cit. 2016-04-26]. Dostupné z: http://testing.bredex.de/jubula-download-page.html

[19] Sikuli. *Sikuli 1.1.0 "SikuliX".* [online]. Launchpad, 2015 [cit. 2016-04-26]. Dostupné z: https://launchpad.net/sikuli/sikulix/1.1.0

[20] HLAVA, Tomáš. Testování softwaru. *Druhy, typy a kategorie testů.* [online]. 2011 [cit. 2016-04-26]. Dostupné z: http://testovanisoftwaru.cz/category/druhy-typy-a-kategorie-testu/

[21] HEROUT, Pavel. *Texty k přednáškám z předmětu KIV/OKS.* Plzeň, 2015.

Přílohy

Příloha A: Podklad pro dobrovolníky - Jubula

Nový test vytvořte pravým kliknutím do boxu „Test Case Browser“ a výběrem možnosti „New. . . “. Nově vytvořený test vhodně pojmenujte a potvrd'te. Nový test otevřete k editaci poklepáním levým tlačítkem myši.

Spouštění vytvořených testů
Testy musejí být uložené. V boxu „Test Suite Browser“ vytvořte nový případ kliknutím na pravé tlačítko myši a vybráním možnosti „New Test Suite“. Nový případ otevřete v editoru poklepáním na levé tlačítko myši. Poté do něj vložte přetažením z boxu „Test Case Browser“ vámi připravený test. Případ uložte.

V nabídce Start v systému Windows ve složce Jubula spust'te AUT agenta kliknutím na „Start AUT Agent“. Po jeho spuštění klikněte v menu Jubuly na ikonku zeleného škrtlého puntíku. Tím se připojí Jubula k agentovi. Dále klikněte na zelenou šipku (připomínající značku používanou pro `PLAY`). Tím se spustí testovaná aplikace. Nyní je vše připravené pro spuštění testu. To proved'te kliknutím na zelenou ikonku šipky v horní části boxu „Test Suite Browser“. Dbejte na to, aby na ploše nebyla před spuštěním testu viditelná žádná jiná aplikace. Důležité je v průběhu testu s počítačem nepracovat.

Test č. 1
Otestujte zobrazení okna s chybovým hlášením po vložení špatného formátu data do sloupce „Date of publishing“. Všechny akce se vybírají přetažením z `unbound_modules_concrete_[8.2]->Actions (basic)`. Dávejte pozor na zaškrtávání objektů, kterých se daná akce týká, v boxu „Component Names“.

- Vložte text ve špatném formátu do řádku č. 4 do sloupce „Date“.
  - `Input via Keyboard->Table->ub_tbl_replaceTextWithCell` -

Spec

  - Doplňte parametry `[a.b.cdef; 4; equals; 4; equals]`.
  - POZOR – v pravém sloupci v boxu „Component Names“ zaškrtněte `nn_nn_tbl`.

- Pro dokončení vkládání data stisknout tlačítko `ENTER`.
  - `Input via Keyboard->Application->Key Combination->ub_app_pressAnyKey_ENTER`
- Počkejte na zobrazení okna s chybovým hlášením.
  - `Wait->Application->Wait for Window->ub_app_waitForWindow`
  - Doplňte parametry `[Input mistake; equals; 7000; 500]`.
- Klikněte na tlačítko `OK`.
  - `Click->ub_grc_clickLeft_single`
  - POZOR – v pravém sloupci v boxu „Component Names“ zaškrtněte `nn_nn_grc`.
- Počkejte na zavření okna s chybovým hlášením.
  - `Wait->Wait for Window to Close->ub_app_waitForWindowToClose`
  - Doplňte parametry `[Input mistake; equals; 15000; 500]`.

V případě plné funkčnosti aplikace, proběhnou všechny kroky bez chyby.

Test č. 2
Vytvořte nový test. Otestujte přidání nového řádku tabulky pomocí menu.

- V menu testované aplikace vyberte položku „Library/Add“.
  - `Select->Menu Bar->ub_mbr_selectEntry_byTextpath`
  - Doplňte parametry `[Library/Add; equals]`.
- Zkuste kliknout na pátý (nový) řádek tabulky.

- `Select->Table->ub_tbl_selectCell`
- Doplňte parametry `[5; equals; 2; equals; 1; 50; percent; 50; percent; no; 1]`.
- POZOR – v pravém sloupci v boxu „Component Names“ zaškrtněte `nn_nn_tbl`.

O funkčnosti přidávání řádku pomocí menu rozhodne druhý krok testu. Funguje-li přidávání řádků správně, povede se nám na nový pátý řádek tabulky kliknout. V opačném případě nastane chyba.

Test č. 3
Vytvořte nový test. Otestujte smazání řádky tabulky pomocí menu.

- Klikněte na třetí řádek tabulky.
- Vyberte v menu možnost „Library/Remove“.
- Klikněte na třetí řádek tabulky.
- Zkontrolujte, zda se ve třetím řádku, druhého sloupce tabulky nachází obsah z (původně) čtvrtého řádku, druhého sloupce. Tím zkontrolujeme, zda se třetí řádek opravdu smazal a na jeho místo se tedy posunul řádek čtvrtý.

Proveďte potřebné úkony pro spuštění testu.

Příloha B: Podklad pro dobrovolníky - SikuliX

SikuliX využívá pro vytváření testů jazyk Python a screeny testované aplikace. Před vytvářením testů musíte nejprve spustit testovanou aplikaci. Aplikace je uložena na ploše pod názvem `Tabulka.jar`. Dbejte na to, aby na ploše nebyla viditelná žádná okna jných aplikací kromě testované aplikace. Všechny funkce nejprve vytvořte a poté použijte. Novou funkci vytvořte následující konstrukcí

```
def Název():
   zde vykonávaný kód
   vše v dané funkci musí být odsazeno tabulátorem
   ukončením odsazení začínáme psát hlavní kód
```

Vytvořenou funkci poté pouze zavolejte pomocí `Název()`. Během volání funkce nezapomeňte vynechat odsazení řádků.

Test č. 1
Otestujte zobrazení okna s chybovou hláškou po vložení špatného formátu data.

- Vytvořte novou funkci s názvem `Date`.
  - `def Date():`
- Nalezněte buňku, ve které budete editovat text. Můžete vytvořit screen buňky manuálně. Druhá možnost je přetažení akce, kterou budete provádět, z levého menu. Tím se přepnete do automatického režimu snímkování.
  - `find(zde screen buňky s datem 22.06.2008)`
- Pro označení testu v buňce na ni dvakrát klikněte levým tlačítkem myši.
  - `doubleClick(zde screen buňky s datem 22.06.2008)`
- Vložte text ve špatném formátu.
  - `type("a.b.cdef")`
- Pro dokončení vkládání textu stiskněte tlačítko `ENTER`.
  - `type(Key.ENTER)`

- Vytvořte vhodnou podmínku ověřující existenci okna s chybovým hlášením.
  - `if exists(zde screen okna s chybovým hlášením):`
    `click(zde screen tlačítka OK v novém okně)`
    `print "Test probehl bez chyby."`
    `else:`
    `print "Test skoncil chybou."`

Test č. 2
Otestujte přidání nového řádku pomocí tlačítka „Add“.

- Vytvořte novou funkci s názvem `Add`.
  - `def Add():`
- Klikněte na tlačítko „Add“.
  - `click(zde screen tlačítka Add)`
- Vytvořte vhodnou podmínku ověřující existenci nově vytvořeného řádku.
  - `if exists(zde screen nové řádky):`
    `print "Test probehl bez chyby."`
    `else:`
    `print "Test skoncil chybou."`

Testovanou aplikaci máte již spuštěnou. Dbejte na to, aby v průběhu testu nebyla viditelná okna jiných aplikací, než naší testované. Mohlo by to negativně ovlivnit průběh testu. Projekt uložte, volte název bez diakritiky. Test spusťte kliknutím na ikonu s černou šipkou „Run“. Za běhu testu s počítačem nepracujte.

Test č. 3
Otestujte smazání řádky pomocí menu.

- Nalezněte třetí řádek tabulky.
- Klikněte na třetí řádek tabulky.
- Klikněte na položku „Library“ v menu.
- Klikněte na položku „Remove“ v menu.
- Vhodnou podmínkou se ujistěte, že třetí řádek tabulky neexistuje.

Test uložte a spusťte.

Příloha C: Obsah přiloženého CD

- Dokumentace
- Projekt v Jubule pracující nad funkční aplikací.
- Projekt v Jubule pracující nad nefunkční aplikací.
- Projekt v SikuliX pracující nad funkční aplikací.
- Projekt v SikuliX pracující nad nefunkční aplikací.
- Testovaná aplikace ve funkční podobě.
- Testovaná aplikace s vytvořenými chybami.