TAMPEREEN
AMMATTIKORKEAKOULU

SELECTING TOOL FOR AUTOMATED TESTING OF USER INTERFACE

Petri Pokela

Opinnäytetyö
Huhtikuu 2016
Tietotekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotekniikka

PETRI POKELA:
Selecting tool for automated testing of user interface

Opinnäytetyö 29 sivua, joista liitteitä 0 sivua
Huhtikuu 2016

Tässä työssä käsitellään regressiotestauksen automatisoinnin aloittamista yrityksessä sekä kuvataan yleisesti testauksen ja Scrum-menetelmän teoriaa. Tämä työ toteutettiin Youredi Oy:lle uuden käyttöliittymän testauksen tehostamiseksi.

Opinnäytetyössä käydään läpi yleistä ohjelmistotestauksen teoriaa ja kuvataan eri testausmenetelmiä ja testaustasoja. Ohjelmistoprojektien nykyaikainen ketterä kehitysmenetelmä Scrum ja siihen liittyvät nopeat julkaisuvälit luovat tarpeen myös jatkuvaan ohjelmistotestaukseen. Se on mahdollista testausautomaation avulla.

Tässä työssä vertaillaan neljää markkinoilla olevaa testausautomaatio työkalua Youredin määrittelemiä vaatimuksia vasten. Vertailu tehdään painoarvotaulukon avulla. Eniten pisteitä saadulla työkalulla toteutetaan automaattitestitapauksia.

Jotta testitapauksia voidaan toteuttaa, tehdään ensin testisuunnitelmat, jotka toteutetaan valitulla työkalulla. Testiajojen raportti toimii apuvälineenä näyttämään ohjelmiston toiminnan ja virheen sattuessa raportti auttaa paikallistamaan virheen sijainnin.

Asiasanat: testaus, testausautomaatio, Scrum, työkalun valinta

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
ICT Engineering
Software engineering

PETRI POKELA:
Selecting tool for automated testing of user interface

Bachelor's thesis 29 pages, appendices 0 pages
April 2016

This thesis handles starting of regression test automation in a company and also describes overall testing theory and Scrum method. This thesis is made for company Youredi Ltd to make user interface testing more efficient.

Software testing theory is introduced and different testing methods and testing types are described in this thesis. Nowadays the modern way to develop software projects is Scrum method, which makes frequent software releases possible. That is why software testing needs to be continuous and it is possible by automating software testing.

Four commercial test automation tools are compared against Youredi’s requirements. Comparison is made by using weightage table. Tool with the best points is used to implement some automated test cases.

Test plan with defined test cases needs first to be written to be able to implement test cases with the selected tool. Test run report indicates if the software is working correctly and in case of an error the report helps to locate where the error occurred.

Key words: testing, test automation, Scrum, tool selection

SISÄLLYS

ABBREVIATIONS AND TERMS

AJAX	Asynchronous JavaScript and XML
AJAX-call	Asynchronous HTTP-request
ASP.NET	Web framework for building Web applications and Web sites by Microsoft
GUI	Graphical user interface
MVC	Software architectural pattern called model-view-controller
PHP	Server scripting language

1 INTRODUCTION

This thesis is made for company Youredi Ltd and consists of selecting and evaluating test automation tool for user interface testing. Youredi Ltd is an international software company offering advanced cloud-based integrated data exchange services.

Youredi software's user interface rewriting was started in autumn 2015 and it arose also a need to make GUI testing more efficient. The software is developed by using Scrum method, which means that new features are included in the software all the time. Fast development cycles require efficient way to make sure that all existing features are still working correctly. Test automation tool was decided to be taken in use to help continuous software validation.

This thesis describes shortly testing methods, testing types, Scrum method, continuous integration, continuous delivery and automated testing. Among plenty of available commercial test automation tools, most suitable tools are searched and selected for test automation tool comparison.

In this thesis evaluation test scripts are created with the most suitable test automation tool. Test scripts are implemented according to the test plans and test cases. When test script run is completed, the test automation tool creates a detailed test run report, which helps developers quickly to indicate, locate and solve the found problems.

2 GENERAL TESTING THEORY

Software testing is one entity of software engineering. Software tester’s job is varied depending on the task he is doing. Tester might have to write code, write documentation or interview test users depending what is happening at the moment. That is why professional software tester’s job might have different tasks at different software company. (Kasurinen J. P. 2013.)

Testing is done to ensure that the developed product is working as it is defined and meant to work. Testing is continuous comparing against the specification and should be part of each product development phase, not only at the end of the project. The earlier the design and implementation can be verified to be correct, the less work later in development is needed to correct the mistakes.

2.1 Basics of software testing

Some of basic software testing methods are blackbox testing, whitebox testing and greybox testing.

2.1.1 Blackbox testing

Black box testing is a testing technique that focuses on the execution of the system and output is generated against any input. In black box testing system gets input and informs the output results, but it ignores the internal mechanism inside the system. Black box testing is a simple mechanism and it can be used at any point of software testing if there is a system that is performing some kind of functionality. (Kasurinen J. P. 2013.)

2.1.2 Whitebox testing

White box testing is a testing technique that focuses on the internal mechanism of the system. System gets inputs and tester can see and knows what happens inside of the system and can track down error information back to its source. White box tests are more deep and accurate than black box tests. (Kasurinen J. P. 2013.)

2.1.3 Greybox testing

Grey box testing technique is a combination of black box and white box testing techniques. Grey box testing can focus on the execution of the system, but its user also sees what happens inside of the system. Basically this is a technique that can check that requirements are filled (model coverage) and also that source code branches have been checked (code coverage). (Kasurinen J. P. 2013.)

2.2 Testing types

There are several testing types like:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing
- Regression Testing

2.2.1 Unit Testing

Unit testing is the testing of an individual module, unit, function or group of related units. Unit testing makes sure that implemented change works as part of the system. Unit testing is usually done by the programmer to test that implementation works as expected with all possible input values. Unit tests can be run automatically after every source code change. (Kasurinen J. P. 2013.)

2.2.2 Integration Testing

Integration testing is testing of combined components which produce output. In integration testing a new component is added to pretested system. If the new component includes couplings that do not yet exist, tester have to build a group of stubs to start the system even the system does not have all the function implementations yet. The main purpose for integration testing is to test that components work together as a system. (Kasurinen J. P. 2013.)

2.2.3 System Testing

System testing ensures that the whole system is working in different environments with real data. System testing is done after all of the components are implemented together at integration testing phase. System testing is done in the testing environment and the main purpose is to find all possible errors before moving to Acceptance testing phase. (Kasurinen J. P. 2013.)

2.2.4 Acceptance Testing

Acceptance testing is done to ensure customer that the delivered product meets the requirements and work as they expected. Acceptance testing is often done and officially accepted as a working product by customer. After acceptance testing product is officially delivered to customer. (Kasurinen J. P. 2013.)

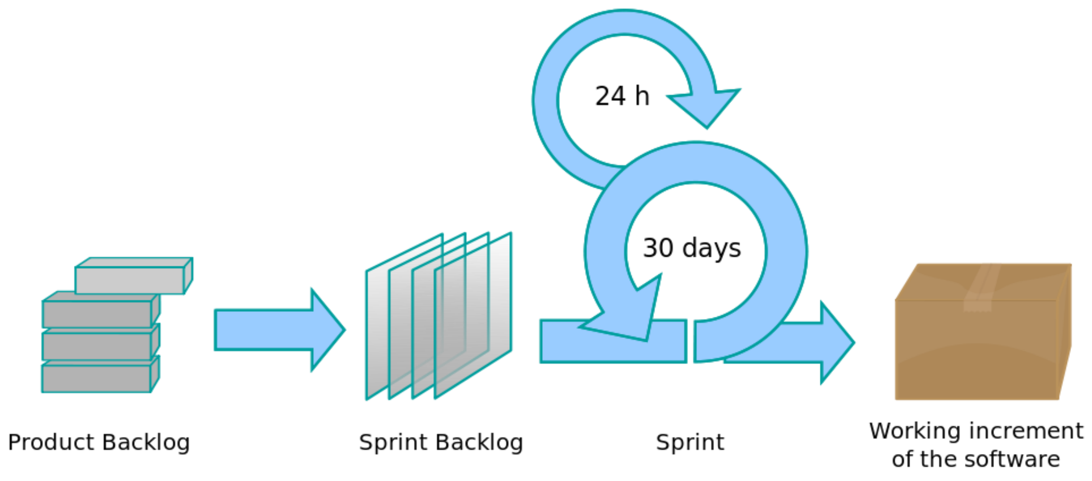
2.2.5 Regression Testing

Regression testing is basically continuously retesting the system. Regression testing is done after modification of the system, component or group of related units to ensure that the product is still working correctly after modifications. The most important part of regression testing is to ensure that there are no errors that have already been fixed before modification, and that the modification does not encounter any new errors on the build. Regression testing is suitable to be automated. (Kasurinen J. P. 2013.)

3 TESTING PROCESS IN PROJECTS

3.1 Scrum method

Scrum is agile method to develop software in smaller steps. The features are incremented one at a time to the application. This makes possible to release application version frequently. Product owner defines the features to be done in one sprint, which is typically 2 weeks to 1 month long. The development team implements the agreed features. Scrum master controls that everything goes smoothly. Following picture describes Scrum method.



PICTURE 1. Scrum method (Scrum picture 2016.)

The Scrum team consists of a Scrum master, a Product Owner and the Development Team. Scrum team is self-organized team working independently by making their own decisions and taking the responsibility of the work. Scrum team is also cross-functional team knowing exactly how to work and do not depend on other teams. “The team model in Scrum is designed to optimize flexibility, creativity, and productivity.” (Scrum Guide 2014.)

3.2 Continuous integration

Different software parts are developed by multiple developers and teams at the same time. To make sure that the whole software is always possible to build and run, the source code should be often saved to source code version controlling system. From there, after every saving, build server verifies the compilation and software package creation. This is called continuous integration. In case of error developer gets immediate feedback from build server and can quickly correct the problem. This technique minimizes integration problems and allows teams to focus producing common source code faster. (Fowler Martin 2006.)

3.3 Automated testing

"Test Automation software is the best way to increase the effectiveness, efficiency and coverage of your software testing." (SmartBear 2016.) Test automation is test method where application testing is done by using test automation tool. Test automation goal is instead of manual testing to run automatically test cases which needs to be repeated continuously. This leaves more time for software engineers to concentrate to the application development. Test automation also helps to verify that source code changes do not affect to already verified existing software functionality.

Automated test scripts can be created for regression tests, which are needed to run repeatedly. Simplest way to do scripts is recording software using and then run tests at any time. Software input value entering can be recorded and the desired output after software function execution can be validated. Scripts are created according to the carefully written test cases, which is also as demanding development work as the software development work itself. Automated test scripts for the GUI can be finalized after the correct product functionality is first verified manually.

When test run is completed, test automation tool generates a test report containing detailed status of the test run. Test report helps developer to locate the malfunction and it is quicker to correct the error. Test report is also an evidence that testing is done.

When same tests are run always automatically, developer/tester can create new test cases instead of manually testing the same things. In the long run this leads to better test coverage of the software.

3.4 Continuous delivery

“Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.” (Fowler Martin 2013.) Continuous delivery is achieved by continuously integrating the software, building executables, and running automated tests on product to detect problems. There is always a production-like environment where executables are pushed to ensure that the software will work in production.

Doing continuous delivery means that software is always deployable throughout its lifecycle and team prioritizes keeping the software deployable all the time. Build server informs developers in case saved source code change makes system unable to build itself. Everyone on the team also has a push-button that performs deployment to any version of the software to any environment on demand.

4 TOOL SELECTION METHOD

To find the best suitable automated testing tool for Youredi needs was done by using a selection process. First of all the thesis team decided what requirements were needed for automated testing tool and how important each requirement is (weightage). After listing the requirements begun the research of the right automated testing tools. At the selection process thesis team decided that four tools will be compared to each other.

Every requirement is given a percentage weightage and the total percentage is 100%. All of the selected tools get points from 1 to 5 depending how well it fills the listed requirement. After every tool has a points in in every cell, the point value is multiplied with the percentage that the requirement has. Finally all multiplied points are summed together to get the total score for each automated testing tool.

Requirements and tool estimation are entered in table structure described in following table.

TABLE 1. Selection table example.

Requirements	Weightage (%)	Tool 1	Tool 2	Tool 3	Tool 4
Reg 1					
Reg 2					
Reg 3					
Total score	100				

5 AUTOMATED TESTING TOOL COMPARISON

There are a lot of tools available for automated testing on the market. Internet searches were initially used to find most suitable automated testing tool for Youredi Ltd. The amount of tools to be compared was limited to four tools from different manufacturers. Each tool was evaluated against the requirement specified with the thesis team. Selected tools for comparison:

- Test Studio from Telerik
- Ranorex from Ranorex
- Selenium from Selenium contributors
- TestComplete from SmartBear

5.1 Tool requirements

Thesis team was specifying following requirements for automated testing tool in meetings:

- Cross browser support
    - Firefox, Chrome etc…
- Possible to record script
    - Recording when manually using the software
- Possible to program scripts
    - Software using is implemented by using programming language
- Possibility to use templates
    - Reusable modules created by programming
- Possibility to control execution according user selections
    - Recognition and execution according to user actions
- Possibility to use object instead of image or coordinates
    - Recognize named object from UI
- Possibility to define generic tests
    - Dynamic using of software by programming automatic inputs
- Possibility to use known starting state
    - Testing is always started at same state for example login page

- Possibility to scale according to browser size (tablet, mobile)
  - UI works on different display sizes without modifications
- Possibility to validate document properties (size, content, visibility)
  - Label, text box, combo box etc. content can be validated
- Possible to validate data-object in memory
  - Variable and data-object content can be validated
- Possibility to validate AJAX-calls (erroneous calls)
  - AJAX-calls content can be validated
- Possible to get test run report
  - Report containing test run result with details
- Deliverer documentation/Help
  - Proper documentation to help user to use the tool
- Deliverer support
  - How easy it is to get support from manufacturer
- Free trial version
  - Possibility to evaluate tool before purchasing it
- Price
  - Purchasing price for the first time
- Annual Price
  - Continuous annual pricing
- Deliverer reliability
  - Company history and references
  - What are the future developing plans of the tool
  - How well known the tool is

Weightage percentage was defined separately for each requirement totally summing to 100 percent.

5.2 Compared tools

This section goes briefly through the selected tools. Section also gives information on how well tool matches with requirements.

5.2.1 Telerik Test Studio

Test Studio is easy-to-use tool for web, desktop and mobile application testing including load testing possibilities. Test Studio makes it possible to test applications on various platforms using different frameworks and tools. It supports following technologies: ASP.NET, AJAX, Silverlight, PHP and MVC. Tests can be recorded and it also allows using programming language to create test scripts. (Telerik Test Studio website 2016.)

Finding detailed information from Telerik Test Studio web pages is not easy. Telerik is not focusing only on test automation tool, which made me wondering if this company concentrates enough on this tool.

5.2.2 Ranorex

Ranorex allows automating web, desktop and mobile application testing. Ranorex supports many browsers and combines different technologies, platforms and devices. Ranorex both records user interactions and plays them back to execute tests. Tests can be recorded and played back. After test run detailed test run report is generated. Ranorex has advanced GUI object recognition, which makes test maintaining easier when object location on UI is moved. Ranorex allows using an API for C# and VB.NET to program own reusable test modules. Ranorex is well known commercial tool to build and run automated web and GUI tests. (Ranorex website 2016.)

Ranorex web pages give professional feeling about the product. On the web pages is testing theory and tips how to start building test automation. Ranorex is fully concentrating on test automation. They also have a large group of big companies using it.

5.2.3 Selenium

Selenium is testing tool to automate web application testing in different browsers. With Selenium tests can be recorded and tests can be run with playback function. It is also possible to write tests by programming them by using languages like Java, C#, etc. The tests can then be run in many web browsers. Test run reports can be built by user itself.

Selenium is open-source software and it is released under the Apache 2.0 license. It can be downloaded and used for free. (Selenium website 2016.)

Selenium seems to be efficient tool but requires advanced software developer skills. There might be longer learning curve to take it in use. There is no official support for the tool, it is based on user groups and chat rooms, which might be a threshold for a new user.

5.2.4 TestComplete

TestComplete can be used to create test automation for web, desktop and mobile applic-tations. Most popular web browsers are supported. Tests can be recorded and played back. After test run detailed test run report is generated. TestComplete supports multiple script-ing languages like C#, VisualBasic and JavaScript, etc. AJAX, HTML forms, Javascript, and Flash content can be tested with different browsers and operating systems. Objects inside web applications can be identified and validated. (TestComplete website 2016.)

TestComplete is professional tool for building software test automation. They have well organized web pages containing online help. Each module must be separately purchased, so the pricing is confusing.

5.3 Tool selection

Following table contains real information like described earlier in table 1.

TABLE 2. Tool comparison

Requirements	Weightage (%)	Ranorex	Telerik Test Studio	Selenium	Test Com-plete
Possible to test Web applications	6,25	5	5	5	5
Cross browser support	3,33	5	5	4	5
Possible to record scripts	2,51	5	5	5	5
Possible to program scripts	2,51	5	5	4	3

Possibility to use templates	3,33	3	3	3	3
Possibility to control execution according user selections	6,25	4	3	2	4
Possibility to use object instead of image or coordinates	6,25	5	5	5	5
Possibility to define generic tests	6,25	3	3	3	2
Possibility to use known starting state	6,25	5	3	2	5
Possibility to scale according to browser size (tablet, mobile)	6,25	4	4	3	4
Possibility to validate document properties (Size, content, visibility)	6,25	5	5	5	5
Possible to validate data-object in memory	6,25				
Possibility to validate AJAX-calls (errorneus calls)	3,33	5	4	4	3
Possible to get test run report	6,25	5	5	3	5
Deliverer documentation/Help	3,33	5	5	3	5
Deliverer support	3,33	5	5	4	5
Free trial version	6,25	5	5	3	5
Price	6,25	4	3	5	4
Annual Price	6,25	4	4	5	5
Deliverer reliability (History, references, future, well known)	3,33	5	5	4	5
Total score	100	424,59	396,26	352,1	412,91

Currently the development team is small in Youredi and it is enough to purchase one test automation license. That is why pricing points do not have more differences between tools. Deliverer support points do not either have more differences, because Selenium was given good points due to their online user group chat rooms even they don't have support service.

All chosen tools in principle could have been selected. Differences were minor and all chosen tools filled Youredi’s needs. Selected tool for Youredi test automation is Ranorex, because it got highest scores during the comparison in table 2.

Original plan was to evaluate two tools with the best total score, but due to organizational reasons only tool with the best total score was evaluated. If the chosen tool does not pass testing phase, then tool with second highest total score can be evaluated. Second tool evaluation is not included in this thesis.

6 TEST PLANNING

Test cases and planning is documented so testing can be repeated any time. Each test case must fulfill following criteria:

- Repeatability: test can any time be similarly repeated.
- Unambiguous: test can be interpreted only one way.
- Traceability: test environment and configuration are traceable.

Each test case contains only one operation to be verified. Test case needs a short, concise title and description. Test case clearly defines the purpose and scope of its operation. Use simple language, so anyone understands it, do not use unnecessary steps or words. Use rather imperative. Test case describes what to do and what is the expected result. There must be enough description about the system behavior, so tester understands it. There must be positive test cases to confirm that software works as it is supposed to work and negative test cases to confirm that software does not do things it shouldn’t. Test plan contains initial setup to start the test.

Test cases are created by developer, tester and product owner. They will always be improved and created more whenever some new case appears. Tests used with automated test scripts contain following chain: known start state → setup → test execution including validation → restore to known state. It is important to return to the known state after the test execution.

6.1 Test cases

This thesis is about selecting and evaluating automated testing tool so only few test cases were created. All of the test cases follow same pattern in documentation. All test cases must have

- ID
- Title
- Pre-Conditions
- Test steps
- Expected results

First test case is called “Login Fail” which basically tests all options how user can fail to login to the system. In the table 3 there are multiple test steps, which are executed separately, but all having same expected result.

TABLE 3. Login Fail

ID	1
Title	Login Fail
Pre-Conditions	Open Youredi Web page
Test Steps	1. Enter incorrect Login ID 2. Enter password 3. Click "Login"
Test Steps	1. Enter correct Login ID 2. Enter incorrect password 3. Click Login
Test Steps	1. Enter empty Login ID 2. Enter password 3. Click Login
Test Steps	1. Enter correct Login ID 2. Enter empty password 3. Click Login
Test Steps	1. Enter empty Login ID 2. Enter empty password 3. Click Login
Test Steps	1. Enter incorrect Login ID 2. Enter empty password 3. Click "Login"
Expected results	Page informs about login failure

When test case should success, it is enough to have one test step. Like in table 4 there is only one test step with several actions and expected result tells user what is going to happen.

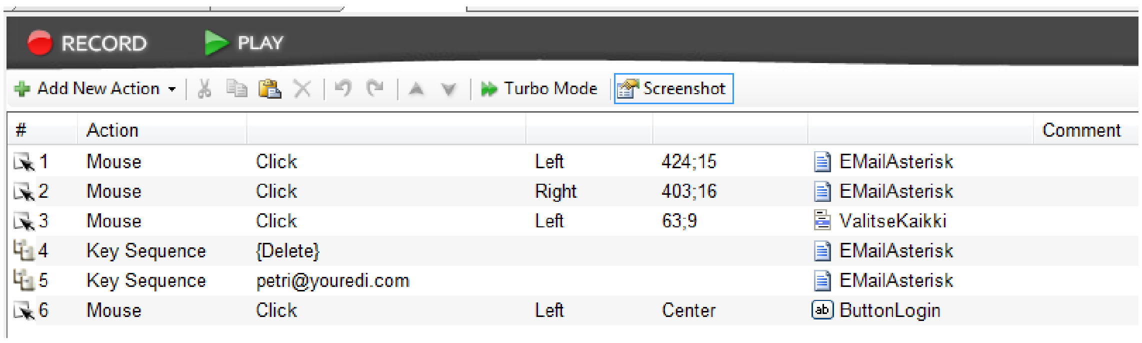
TABLE 4. Login Success

ID	2
Title	Login Success
Pre-Conditions	Open Youredi Web page
Test Steps	1. Enter correct Login ID 2. Enter correct password 3. Click "Login"
Expected results	User is transferred to main page with his user

7 TEST SCRIPT IMPLEMENTATION

7.1 Test case recording

Simplest way to start doing automated test scripts is to record software using. Ranorex saves all user actions and expected result validation can also be recorded.

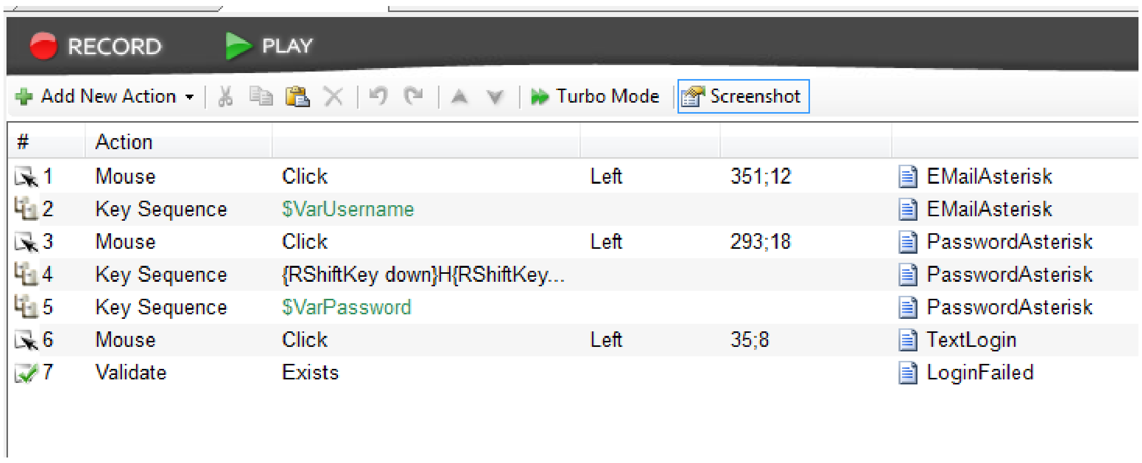


#	Action					Comment
1	Mouse	Click	Left	424;15	EMailAsterisk	
2	Mouse	Click	Right	403;16	EMailAsterisk	
3	Mouse	Click	Left	63;9	ValitseKaikki	
4	Key Sequence	{Delete}			EMailAsterisk	
5	Key Sequence	petri@youredi.com			EMailAsterisk	
6	Mouse	Click	Left	Center	ButtonLogin	

PICTURE 2. Login Succeeded Recording

Picture 2 contains login succeeded recording. Each mouse click, data entering is indicated as own row in the script. User name is entered here as plain text.

Picture 3 contains login failed recording. Here the user name and the password is entered as variables. This is more flexible way to enter data to test scripts.

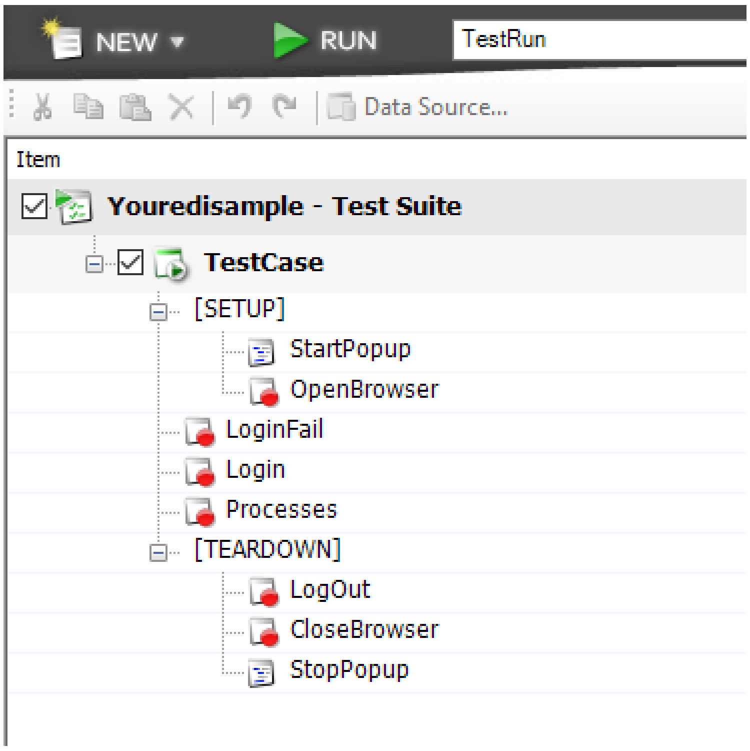


#	Action				
1	Mouse	Click	Left	351;12	EMailAsterisk
2	Key Sequence	$VarUsername			EMailAsterisk
3	Mouse	Click	Left	293;18	PasswordAsterisk
4	Key Sequence	{RShiftKey down}H{RShiftKey...			PasswordAsterisk
5	Key Sequence	$VarPassword			PasswordAsterisk
6	Mouse	Click	Left	35;8	TextLogin
7	Validate	Exists			LoginFailed

PICTURE 3. Login Failed Recording

7.2 Test case programming

In this thesis setup and teardown sections were implemented by coding. Setup section prepares software for using. Teardown finalizes and closes software in controlled way, so next test can be started from known state. Picture 4 shows the test case tree containing setup and teardown section details.

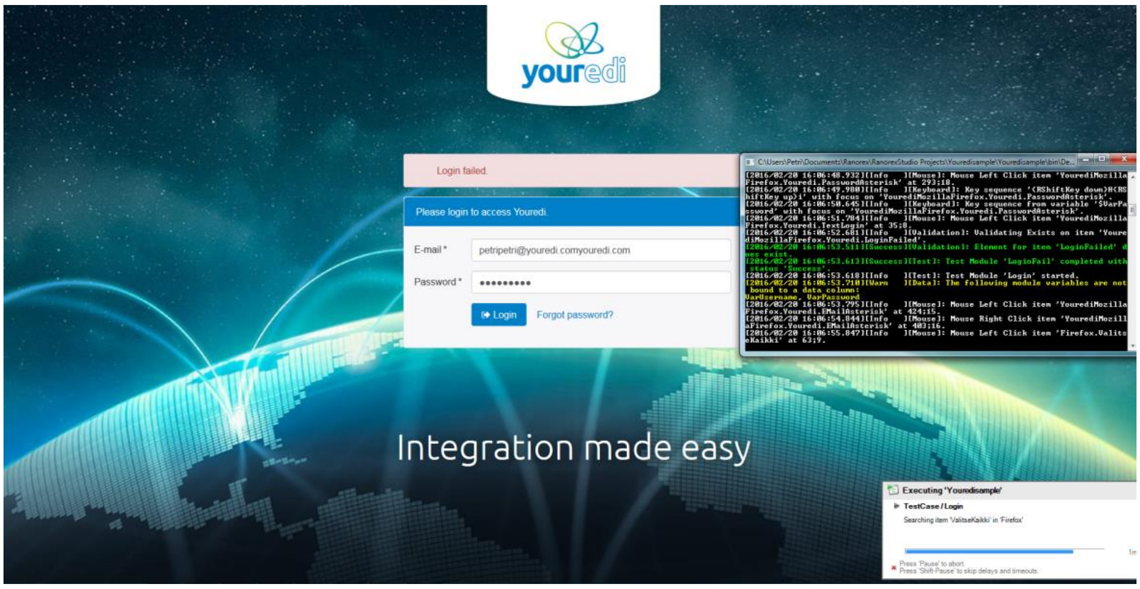


```
Youredisample - Test Suite
  TestCase
    [SETUP]
      StartPopup
      OpenBrowser
    LoginFail
    Login
    Processes
    [TEARDOWN]
      LogOut
      CloseBrowser
      StopPopup
```

PICTURE 4. Setup and teardown sections

7.3 Test script running

Picture 5 has screen capture of expected result described in table 3. It also contains Ranorex command prompt running at background.



```
C:\Users\Petri\Documents\Ranorex\RanorexStudio Projects\Youredisample\Youredisample\bin\De...
[2016/02/20 16:06:48.932][Info    ][Mouse]: Mouse Left Click item 'YourediMozilla
Firefox.Youredi.PasswordAsterisk' at 293;18.
[2016/02/20 16:06:49.980][Info    ][Keyboard]: Key sequence '{RShiftKey down}H{RS
hiftKey up}i' with focus on 'YourediMozillaFirefox.Youredi.PasswordAsterisk'.
[2016/02/20 16:06:50.645][Info    ][Keyboard]: Key sequence from variable '$VarPa
ssword' with focus on 'YourediMozillaFirefox.Youredi.PasswordAsterisk'.
[2016/02/20 16:06:51.784][Info    ][Mouse]: Mouse Left Click item 'YourediMozilla
Firefox.Youredi.TextLogin' at 35;8.
[2016/02/20 16:06:52.681][Info    ][Validation]: Validating Exists on item 'Youre
diMozillaFirefox.Youredi.LoginFailed'.
[2016/02/20 16:06:53.511][Success][Validation]: Element for item 'LoginFailed' d
oes exist.
[2016/02/20 16:06:53.613][Success][Test]: Test Module 'LoginFail' completed with
 status 'Success'.
[2016/02/20 16:06:53.618][Info    ][Test]: Test Module 'Login' started.
[2016/02/20 16:06:53.710][Warn    ][Data]: The following module variables are not
 bound to a data column:
VarUsername, VarPassword
[2016/02/20 16:06:53.795][Info    ][Mouse]: Mouse Left Click item 'YourediMozilla
Firefox.Youredi.EMailAsterisk' at 424;15.
[2016/02/20 16:06:54.844][Info    ][Mouse]: Mouse Right Click item 'YourediMozill
aFirefox.Youredi.EMailAsterisk' at 403;16.
[2016/02/20 16:06:55.847][Info    ][Mouse]: Mouse Left Click item 'Firefox.Valits
eKaikki' at 63;9.
```

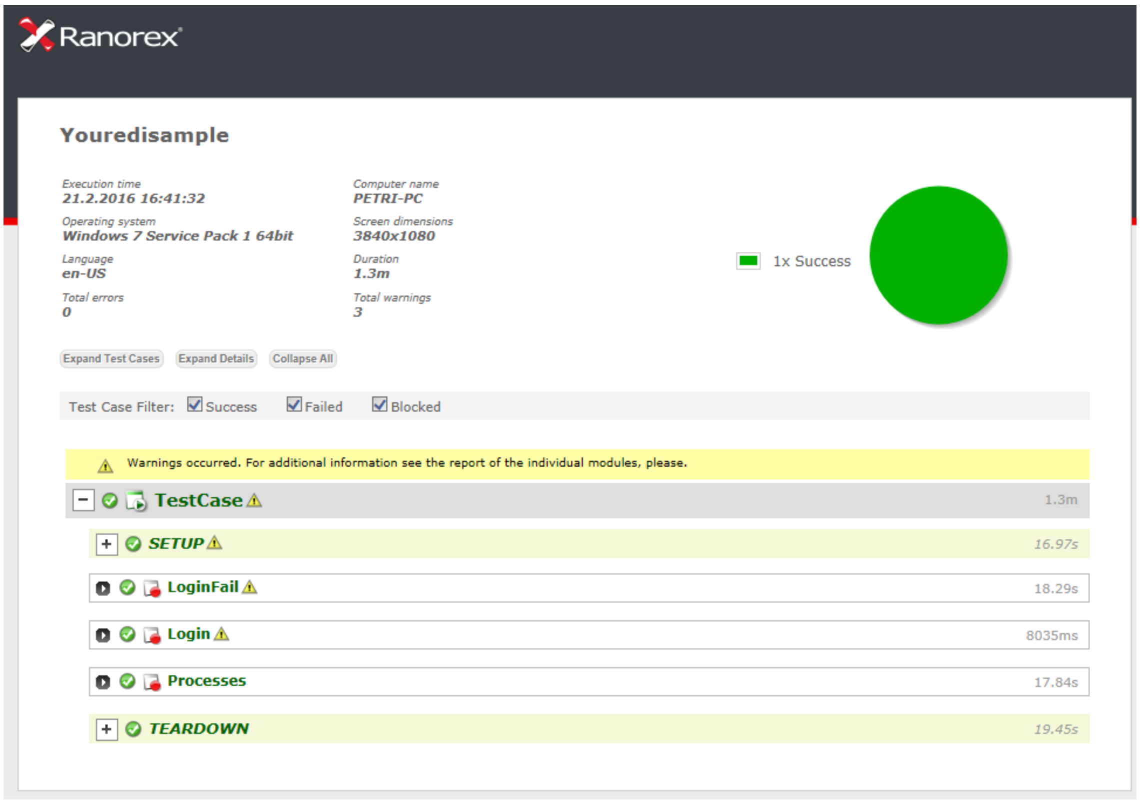
```
Executing 'Youredisample'
TestCase / Login
Searching item 'ValitseKaikki' in 'Firefox'
Press 'Pause' to abort.
Press 'Shift-Pause' to skip delays and timeouts.
```

PICTURE 5. Login fail run and Ranorex command prompt

When test script is running it starts Ranorex command prompt from where it is possible to follow the test run progress. Also the recorded or scripted software using is played back and real UI progress can be followed. This is the part where the savings of automated testing are earned compared to manual testing.

7.4 Test run reporting

When test run is ready, Ranorex creates detailed test report of the test run. Every test case success is separately indicated. Report contains test run time stamp, test environment, test duration and other informative data. When viewing the test run report in Ranorex development environment, from each step user can easily jump to corresponding test step in Ranorex recording or source code.



Ranorex

Youredisample

Execution time
21.2.2016 16:41:32

Operating system
Windows 7 Service Pack 1 64bit

Language
en-US

Total errors
0

Computer name
PETRI-PC

Screen dimensions
3840x1080

Duration
1.3m

Total warnings
3

🟩 1x Success

Expand Test Cases | Expand Details | Collapse All

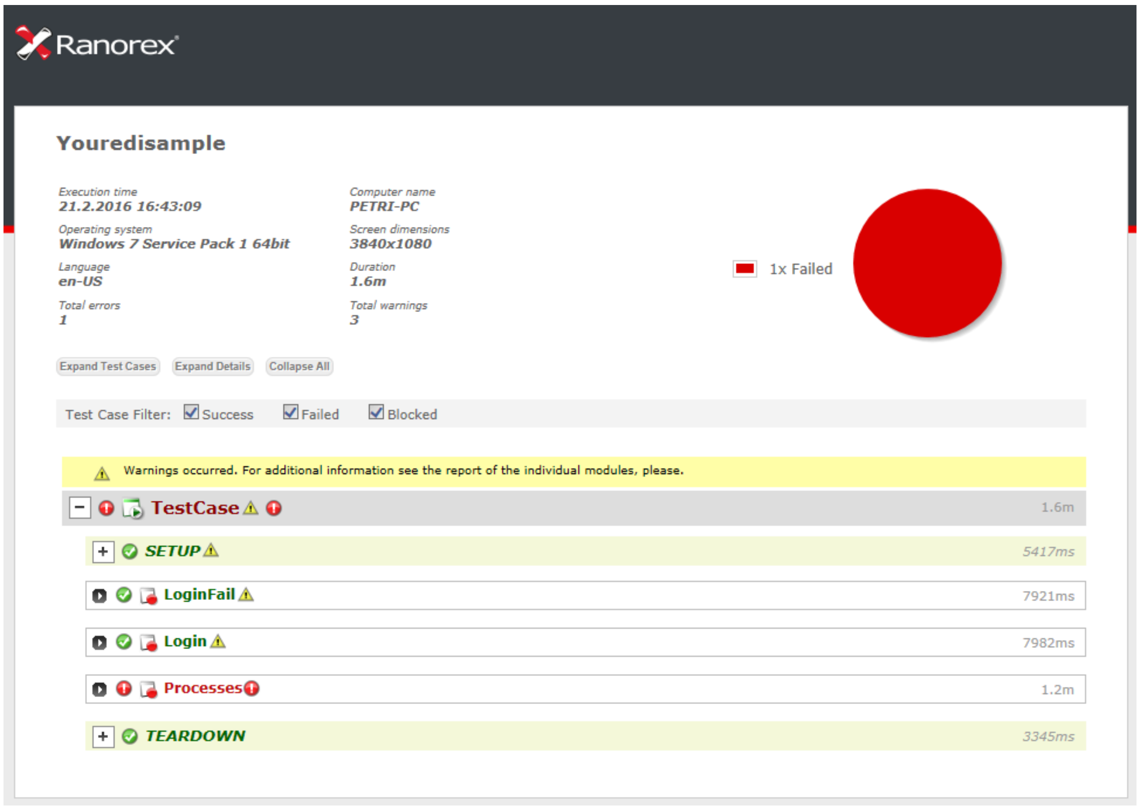
Test Case Filter: ✓ Success ✓ Failed ✓ Blocked

Warnings occurred. For additional information see the report of the individual modules, please.

TestCase	1.3m
SETUP	16.97s
LoginFail	18.29s
Login	8035ms
Processes	17.84s
TEARDOWN	19.45s

PICTURE 6. Succeeded test run

Picture 6 contains fully succeeded test run report. The big green circle on the top of the test report indicates that all test cases are passed. Each test case recording is indicated with icon having small red circle and the small green circle in front of test case indicates, that the test case is passed. Each test case is processed and reported separately. In the beginning of the test setup section is run and at the end teardown section is run.


Youredisample

Execution time
21.2.2016 16:43:09
Operating system
Windows 7 Service Pack 1 64bit
Language
en-US
Total errors
1

Computer name
PETRI-PC
Screen dimensions
3840x1080
Duration
1.6m
Total warnings
3

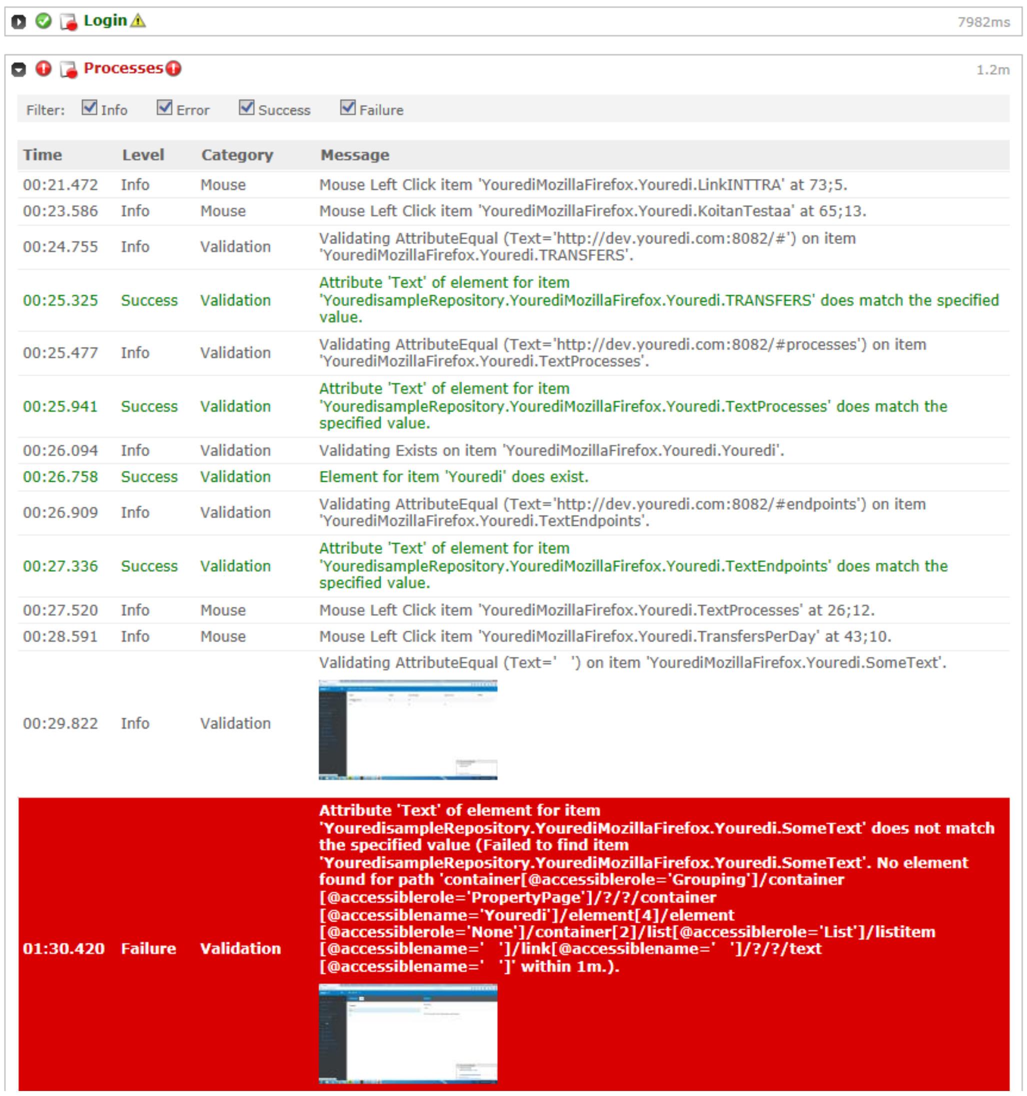
🟥 1x Failed

Warnings occurred. For additional information see the report of the individual modules, please.

TestCase	1.6m
SETUP	5417ms
LoginFail	7921ms
Login	7982ms
Processes	1.2m
TEARDOWN	3345ms

PICTURE 7. Failed test run

If some of the test cases fail during test run, test report will have big red circle on the top of the test report indicating failed test run. Small red circle in front of the test case indicates failed test case. Even test case fails somewhere, teardown section makes sure that software is left to known state after test run like seen in picture 7.



Login 7982ms

Processes 1.2m

Filter: ✓ Info ✓ Error ✓ Success ✓ Failure

Time	Level	Category	Message
00:21.472	Info	Mouse	Mouse Left Click item 'YourediMozillaFirefox.Youredi.LinkINTTRA' at 73;5.
00:23.586	Info	Mouse	Mouse Left Click item 'YourediMozillaFirefox.Youredi.KoitanTestaa' at 65;13.
00:24.755	Info	Validation	Validating AttributeEqual (Text='http://dev.youredi.com:8082/#') on item 'YourediMozillaFirefox.Youredi.TRANSFERS'.
00:25.325	Success	Validation	Attribute 'Text' of element for item 'YouredisampleRepository.YourediMozillaFirefox.Youredi.TRANSFERS' does match the specified value.
00:25.477	Info	Validation	Validating AttributeEqual (Text='http://dev.youredi.com:8082/#processes') on item 'YourediMozillaFirefox.Youredi.TextProcesses'.
00:25.941	Success	Validation	Attribute 'Text' of element for item 'YouredisampleRepository.YourediMozillaFirefox.Youredi.TextProcesses' does match the specified value.
00:26.094	Info	Validation	Validating Exists on item 'YourediMozillaFirefox.Youredi.Youredi'.
00:26.758	Success	Validation	Element for item 'Youredi' does exist.
00:26.909	Info	Validation	Validating AttributeEqual (Text='http://dev.youredi.com:8082/#endpoints') on item 'YourediMozillaFirefox.Youredi.TextEndpoints'.
00:27.336	Success	Validation	Attribute 'Text' of element for item 'YouredisampleRepository.YourediMozillaFirefox.Youredi.TextEndpoints' does match the specified value.
00:27.520	Info	Mouse	Mouse Left Click item 'YourediMozillaFirefox.Youredi.TextProcesses' at 26;12.
00:28.591	Info	Mouse	Mouse Left Click item 'YourediMozillaFirefox.Youredi.TransfersPerDay' at 43;10.
00:29.822	Info	Validation	Validating AttributeEqual (Text=' ') on item 'YourediMozillaFirefox.Youredi.SomeText'.
01:30.420	Failure	Validation	Attribute 'Text' of element for item 'YouredisampleRepository.YourediMozillaFirefox.Youredi.SomeText' does not match the specified value (Failed to find item 'YouredisampleRepository.YourediMozillaFirefox.Youredi.SomeText'. No element found for path 'container[@accessiblerole='Grouping']/container[@accessiblerole='PropertyPage']/?/?/container[@accessiblename='Youredi']/element[4]/element[@accessiblerole='None']/container[2]/list[@accessiblerole='List']/listitem[@accessiblename=' ']/link[@accessiblename=' ']/?/?/text[@accessiblename=' ']' within 1m.).

PICTURE 8. Failed test run expanded

Picture 8 contains expanded failed test case. Ranorex contains error details and also indicates the error location in the test case recording. Now the developer can locate the reason and solve the problem.

8 CONCLUSION

At the moment software solutions are developed by Scrum method, which makes it possible to produce continuous releases, so software must be tested efficiently. Efficiency is possible to achieve only by automating the software testing. During this thesis selecting right tools for automated testing of user interface was complicated task, because currently there are plenty of tools available for automated software testing.

First of all the requirements for automated testing tool were defined so right automated testing tool meeting the requirements was even possible to select. In my opinion the method used to compare the tools against the requirements was excellent. In this thesis Ranorex was calculated to be most suitable tool for Youredi Ltd needs according to the requirement comparison table. According to made evaluation it was a good choice.

Test automation should be a parallel development phase with the software development phase itself. Testing must be carefully planned and test cases according to software requirement should be written. If tests are just recorded without any plan, it will very soon lead to unmaintainable test scripts.

Originally it was meant to evaluate at least two automated testing tools in this thesis, but because of the employment changes done in Youredi Ltd, I had time and possibility to make only basic test scripts with one tool. That's why in this thesis more weight was put on describing software testing and tool selection than implementing the test scripts. During my Ranorex tool trial period there were not yet that much new user interface ready in Youredi’s software.

REFERENCES

Fowler Martin. 2006. Continuous Integration. Read: 10.03.2016. http://www.martin-fowler.com/articles/continuousIntegration.html

Fowler Martin. 2013. ContinuousDelivery. Read: 07.03.2016. http://martin-fowler.com/bliki/ContinuousDelivery.html

Kasurinen Jussi Pekka. 2013. Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo.

Telerik Test Studio. Telerik Test Studio website. Read: 03.03.2016. http://www.tele-rik.com/teststudio

Ranorex. Ranorex website. Read: 03.03.2016. http://www.ranorex.com/

Selenium. Selenium website. Read: 03.03.2016. http://docs.seleniumhq.org/

TestComplete. TestComplete website. Read: 03.03.2016. https://smartbear.com/prod-uct/testcomplete/web-module/overview/

Scrum picture. 2016. Wikipedia. Read: 07.03.2016. https://fi.wikipe-dia.org/wiki/Scrum#/media/File:Scrum_process.svg

Scrum Guide. 2014. ScrumGuides. Read: 07.03.2016. http://www.Scrum-guides.org/Scrum-guide.html

SmartBear. 2016. Why Automated Testing?. Read 11.3.2016. https://smart-bear.com/learn/automated-testing/