Automated Web Application Testing Using Improvement Selenium

by

Sharifah Anis Syed Naser

Dissertation submitted in partial fulfillment of the
requirements for the

Bachelor of Technology (Hons)
(Information and Communication Technology)

AUGUST 2012

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Automated Web Application Testing Using Improvement Selenium

by

Sharifah Anis Syed Naser

A project dissertation submitted to the

Information and Communication Technology Programme

Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

Bachelor of Technology (Hons)

(Information and Communication Technology)

Approved by,

____

DR. SHUIB B BASRI

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

August 2012

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

____

SHARIFAH ANIS SYED NASER

ABSTRACT

The basis of this project is to develop an automated web application testing tool using the enhancement of Selenium which is an open source testing tool. The tool should able to handle automated tests with a group of testers. The current testing tools are proprietary tool and not all test engineers expert or have a deep knowledge in programming which is required to create automated test case. The main problem in testing is time consuming where some of testers are still using manual testing. The scope of study is to develop a tool for software tester who focuses on functional testing and the tool can execute test cases on different platforms which are Windows, Linux and Mac. Each platform has different browsers which are Firefox, Internet Explorer and Chrome.

A qualitative research is done in this project by interviewing several software test engineers who are familiar with automated testing and a framework has been designed in order to fulfill the requirements and to cater the problems that are currently facing. This framework is developed and coded by using Eclipse IDE, Apache ANT and TestNG. As a result, this testing tool is using the combination of Selenium Grid and TestNG in order to create a template for software tester to use it easily. This framework is compatible to all browsers and platforms that are commonly used by Internet users.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1 INTRODUCTION

1.1 Project Background

Agile testing is a software testing practice for projects using agile methodologies. Agile testing involves testing from the customer perspective as early as possible, testing early and often as code becomes available and stable enough, since working increments of the software are released often in agile software development. By using automated script to cut the cost and accelerate the whole test execution process is commonly done in agile testing. Agile testing is developed based on the theory that testers need to adapt to rapid deployment cycles and changes in testing patterns.

All software are developed and designed to meet and fulfill several functional requirements. A functional requirement can be technical, process or business based. Functional testing is the process by which expected behavior of an application can be tested. Performing functional test for the application early in the software development cycle accelerate development, improves quality and reduces risks towards the end of the cycle. Mainly software goes through changes over a period of time. During the course of a single release, or across multiple releases, these changes may arise. Any changes to the software increase the chance of errors of bugs being introduce that breaks existing functionality is not broken is called regression testing.

Test automation is frequently becoming a must in software projects which for the organizations that are using some form of agile methodology. Test automation is defined as using a software tool to run repeatable tests against the application to be tested. Test automation provides responsiveness in regression testing. There are many advantages using test automation and typically are related to the repeatability of the tests and the tests' speed that can be executed. Test automation supports numerous regression testing, quick feedback to developers, virtually unlimited iterations of test case execution and finding defects missed by manual testing.

1.2 Problem Statement

In agile testing, performing a manual testing is kind of tedious and time consuming for a tester to do manually input data and trigger the actions. It requires a lot of human resources and it is impossible for the tester to test every software feature before it is released especially during regression testing. Besides that, the usage of proprietary tool in a company needs a lot of money to purchase and to maintain it such as paying for license renewal every year. Several of the proprietary tools have limited features for the users. A better web application is the compatibility to all Internet browsers that commonly used by Internet users. However, not all testing tools support all type of browsers such as Selenium IDE which only able to record and playback tests on Firefox instead of other browsers. For a tester who is not expert in programming will have some difficulties in creating test cases where there is no test case template to use. The testing tool will takes some time to use it for a group of testers because they have to create the test case from scratch. When there are many test cases that need to be run in a short time, the performance of machine usage will be low. Thus, the test execution process will be slow.

1.3 Objectives

The objectives in this project:-

- To emphasize test as early as possible with less cost of test execution process.
- To handle automated tests with a group of testers.
- To develop an automated web application testing tool for all browsers version and platforms.

1.4 Scope of Study

The scope of study in this project:-

- Target users – software testers.
- Type of browsers – Firefox, Internet Explorer and Chrome.
- Type of platforms – Windows, Linux and Mac.
- Type of testing – functional testing.

CHAPTER 2

LITERATURE REVIEW

2 LITERATURE REVIEW

2.1 Test Automation

Bolton (2010) claims that one important tool or rapid testing is the discipline of exploratory testing – essentially a testing martial art. Exploratory testing combines test design, test execution, test result interpretation, and learning into a simultaneous, seamless process that reveals important information about the product and finds a lot of problems quickly. Test automation make the testing process becomes more effective when it able to reproduce errors and tedious routine tests can be executed without human involvement. Testing will become faster and more efficient in numerous sub-areas like regression, stress, configuration and multi-platform compatibility testing. By implementing test automation, product will become reliable where it improves performance and requirements definition.

Random input generation delivers the best results when combined with automated, due to the numerous and untargeted tests that it produces. For a human it would be tedious to go through these tests, of which only a small proportion reveal faults (Ciupa, et al., 2008).

According to Ciupa et al.(2008), performing manual testing can make the tester become exhausted because they have to test many test cases and do it repeatable. When the tester become exhausted, they will have less creativity to test to find bugs in the system. As the result, the number of faults that the tester found are small.

Mehaffy, et al. (2010) come out with another statement; Quality Assurance (QA) managers are realizing a greater need for test automation with the dynamic and unpredictable nature of the testing process and the need to minimize test schedules. However, not every manual test can be automated such as usability test and security test.

Ndem, et al. (2011) had mentioned in order to automatically test large and complex software systems, a well defined test process and a huge amount of test data are needed. When it comes to test automation, the quality of test data always plays a significant role by reducing the cost and the time required for the test activities. The quality and depth of test cases increase considerably when testers are freed from boring and repetitive tasks such as regression testing. Berner, et al. (2005) supported saying that they have more time to design more or better test cases, focusing on areas that have been neglected so far.

2.2 Selenium

In order to support test automation, Selenium is the best tool that can be used which is a suite of tools to automate web browsers across many platforms. Selenium is an open source tool can runs in many type of browsers and platforms that are commonly used by internet users. It can be controlled by many testing frameworks and programming languages (e.g Java, Ruby, Python, C#, Perl, PHP and Groovy). Besides that, it composed of multiple software tools and each tool has a specific role.

- Selenium 2 (aka. Selenium Webdriver)

  Selenium 2 is the enhancement of Selenium 1 which accepts commands and sends them to a browser. This is done through a browser-specific browser driver, which sends commands to a browser, and retrieves results. Selenium 2 executed tests without a special server like Selenium 1 but it directly starts a browser instance and controls it.

- Selenium 1 (aka. Selenium RC or Remote Control)

  Selenium 1 is a server that accepts commands for the browser via HTTP. When launching html test case, a new instance of Selenium server is needed. It means that the port cannot be same for each parallel run.

- Selenium IDE

  Selenium IDE is a Firefox extension that allows recording, editing and debugging tests in a short time. Previously it was known as Selenium Recorder.

- Selenium Grid

Selenium Grid is a server that allows tests to use web browser instances running on remote machines. One server acts as the hub and it has a list of servers that provide access to browser instance (nodes), and lets tests use these instances. Selenium Grid supports to run tests in parallel on multiple machines, and to control different browser versions.

Fahrurazi (2011) found that not all automation tools can support all browsers versions and OS which are commonly used by internet users. In his research he state that Selenium IDE only support Firefox browsers instead of other available browsers. All softwares developed by software developers are compatible to all browsers because they have to consider that not all internet users using same browsers. Thus, the testing tools also should have functionality that can support all type of browsers to test the compatibility of softwares developed with the browsers used.

Kongsli (2007) had done his review about Selenium and he said that Selenium is a tool for creating and running automated web tests and is a good fit for agile projects where it can be used for creating acceptance tests corresponding to the web application's user stories. The tester can run the test cases just one click and it can save more time and human labor.

Selenium has provided more-efficient ways to handle waiting time for loading page on Ajax application, (Yu & Chao, 2010). In Ajax application, a user has to wait for a while as the asynchronous call completes once he landed on the Ajax web page. To overcome this problem, Selenium will stop waiting if the condition is fulfilled, or return a timeout exception if it exceeds the maximum waiting time and the test case will fail.

Writing web applications is difficult enough, and just when a tester think he is done, he inevitably run into some obscure bug that is only present on one browser. Unfortunately, a unit test does not catch this type of bug. Selenium is the answer to this problem because tester can write functional tests that run in each browser, and then implement some form of continuous integration systems that runs the functional tests upon each the source code, (Gift, 2009). By doing this way, tester able to catch the potential bugs in the browser quickly.

Selenium RC looks like able to support enough parallelism that any additional distributed processing capability would not be needed. In practice, it is not advisable to run more than six browsers on the same Selenium RC server due to performance issues. Besides that, monitoring a large number of Selenium RC servers is a major headache and does not scale very well. This is where Selenium Grid can help to solve those problems (Sirotkin. 2010).

2.3 Maintenance and Reuse of Testing Tool

Programming skills are vital in automation where the tester need to understand the problem or errors found and debug it in the system when they find the problems. Maintainable and reusable tool building requires good programming skills and knowledge of the tool (Loreto, 2006). Loreto try to explain that automation requires technical skills where in software testing nowadays, still have testers who are not expert in programming skills. This problem can be a great challenge to an organization implementing agile methodology in their software projects. While, Sztipanovits, et al.,(2008) agreed with Loreto statement that currently the time and labor cost of grading of tester web application assignments is a shared issue for all related programming courses. Horwath (2007) come out with his support statement saying that most software testing projects do fail because one of the reasons is test technicians with improper skills are assigned to use these automated test tools. Users of these tools must have strong test mentalities and in all but a few situations they must also possess solid programming skills with the automation tool's scripting language.

In this era, all systems are developed as an automated due to save more time and cut the cost of production. Thus, having a great programming skills are required in order to fully utilize automated testing. For this project, the type of programming skills that will be used is Java which is easy to use and easy to find reference if any problems happened. Also, a template of test case which contains the basic line of codes that is useful for tester to use in their project testing. Tester can edit or modify the template according to what they want. From here, tester can learn and improve their programming skills slowly by modifying test case.

Automation will consume more machine resources that will cause the low performance of test execution process. Automation is fully working with machine where it requires less human resources (Razak, 2011). Based on Razak's research, it can say that automated needs human resources just to control or monitor the automation works. Candea, et al., (2010) also support with Razak statement that software testers are still not ready to handle real-sized software (1 million lines of code or more), mainly due to high CPU and memory requirements. The main competitive metric to measure the reliability of a software product is often performance and functionality.

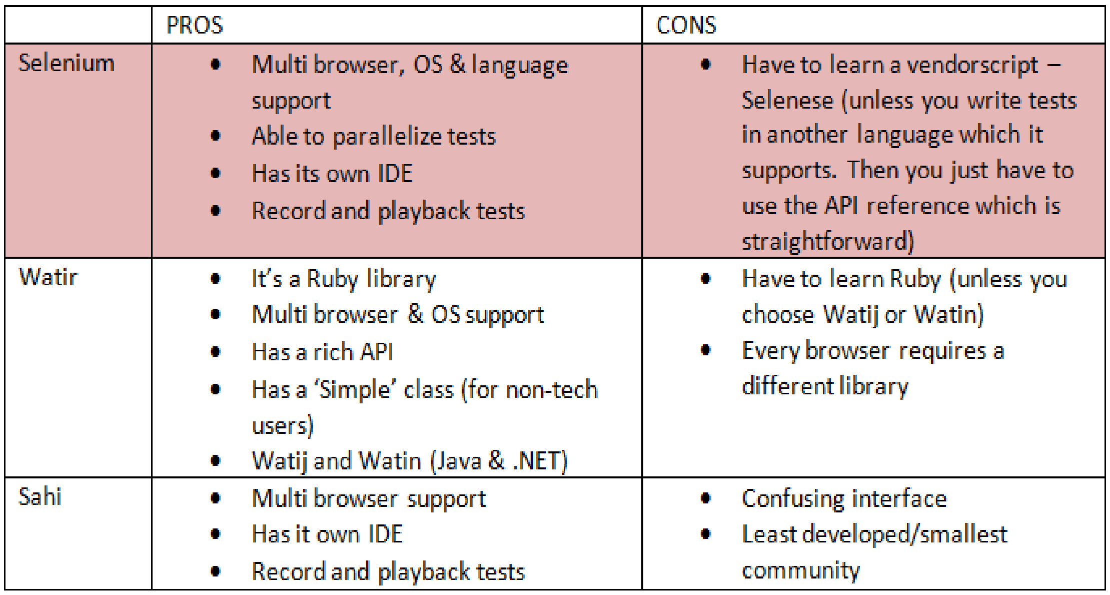
Existing automated techniques, like model checking and symbolic execution are highly effective (Cadar, et al., 2008) but their implementation in industrial general purpose software testing has been limited. There is another support statement mentioned that the limitations are the three challenges faced by automated testing: scalability, applicability and usability (Bucur, et al., 2011). Scalability in automated testing is able to enhance the performance of test execution process. The loads of test cases can be distributed to other machines and make it running parallel at the same time.

It is good when a human just do nothing except monitoring the automation works because it can save more time instead using manual testing which requires a lot of human resources. However, machine resources have their own limits. Sometimes, it can cause a problem to human where the computer hangs while running test execution. Each iteration in agile testing has many test cases that need to be tested. So it is impossible to run all test cases in one computer. Due to avoid that problem happened, all loads (test cases) should be distributed to other machine in order to increase the test execution process performance.

The advantage of using open source tools is it count less maintenance costs because the users does not tied to anyone that need to renew license every year (Macnaghten, 2005). Macnaghten consider a large team of people who would happily maintain the software at minimal cost and he found that it is a foundation for a business model.

A big organization would have big advantage for this issue because they have to allocate their budget carefully that can gain high profit. If they use open source tools in their software project, it can reduce the cost of production and gain high profit. It is true that proprietary tools are better than open source tools but people can modify or enhance the open source tool without using any cash.

From the research done, this project will use Selenium for the automated testing tool. However, several enhancements will do in order to improve the functionality of Selenium. **Table 1** shows the comparisons between Selenium with others open source tools which are Selenium, Watir and Sahi (Raman, 2006).



	PROS	CONS
Selenium	• Multi browser, OS & language support • Able to parallelize tests • Has its own IDE • Record and playback tests	• Have to learn a vendorscript – Selenese (unless you write tests in another language which it supports. Then you just have to use the API reference which is straightforward)
Watir	• It's a Ruby library • Multi browser & OS support • Has a rich API • Has a 'Simple' class (for non-tech users) • Watij and Watin (Java & .NET)	• Have to learn Ruby (unless you choose Watij or Watin) • Every browser requires a different library
Sahi	• Multi browser support • Has it own IDE • Record and playback tests	• Confusing interface • Least developed/smallest community

Table 1: Comparisons between others open source tools

Watir is another web application testing tool similar with Selenium but it is in Ruby programming language. The second tool is Sahi which can be used in Sahi Script, Java, Ruby Sahi Script languages. Both tools have limited programming language support compared to Selenium. According to **Table 1**, these three types of open source tools have one similarity which is multi browser support. However, only Selenium and Watir are able to support multiplatform. Selenium is able to parallelize tests while Watir and Sahi do not have this ability. Sahi is difficult to use that Selenium because it needs installation to start and estimated start time 10 to 30 minutes, depending on Java installation.

By using Selenium as the automated testing tool, there are two types of unit testing framework for Java language that can be used which are TestNG and JUnit (Glover, 2006). Besides that, Selenium Grid which is one of software tools under Selenium will be used in this project. There are few differences between Selenium RC and Selenium Grid (Li & Sun, 2011). As a result, Selenium Grid with TestNG will be used for this project. In **Table 2** will explain why Selenium Grid with TestNG are choose.

Techniques	TestNG	JUnit	Selenium Grid	Selenium RC
Differences	• Support parallel execution (multithreading)	• Not support parallel execution	• Running tests cases parallelly across multiple machines or parallelly on a single machine	• Run test cases across multiple browsers on a single machine

Table 2: Differences techniques

CHAPTER 3

METHODOLOGY

3 METHODOLOGY

3.1 Study Methodology

An interview is done with several software test engineers who are expert in automated testing. These software test engineers are specialized in functional testing thus; they are the main testers who will test the software to ensure whether the product is good or bad. They have been working for a long period in a well known software development company which implementing Agile software development. Purpose of interviewing is to collect data which will be the key to the analysis phase.

From the interview sessions that have been done, some of the testers are still using manual testing in certain testing. The reason why they are still using manual testing is not all of them are very expert in programming which is required in creating automated scripts. Most of them are familiar and know about Selenium but not all of them have a deep knowledge. Some of them either only know the basic techniques of using Selenium or do not know how to use Selenium. Moreover, they realize that using manual testing is consuming more time and cost. However, they are hoping that there will be an easy way for test engineers to do automated testing by using Selenium besides reduces time and cost consuming.

Figure 1 is the framework design for this project after analyzing all gathering data. This framework is used in development phase.

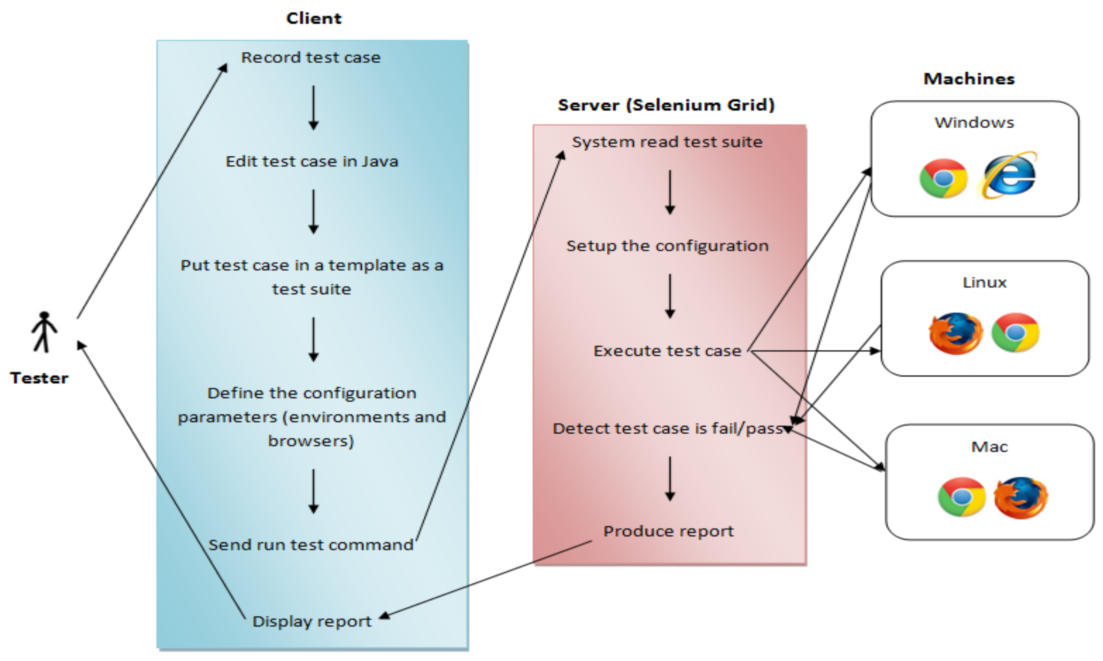


Figure 1: Framework design

The format of test case that is used in this project is Selenium Webdriver in Java language. In this project, the number of browsers used is two types per platforms. There is no limitation for the number of test cases that will be used during test execution.

3.2 Development Methodology

Figure 2 shows the development methodology that implemented in this project. This methodology is adopted from Waterfall methodology.

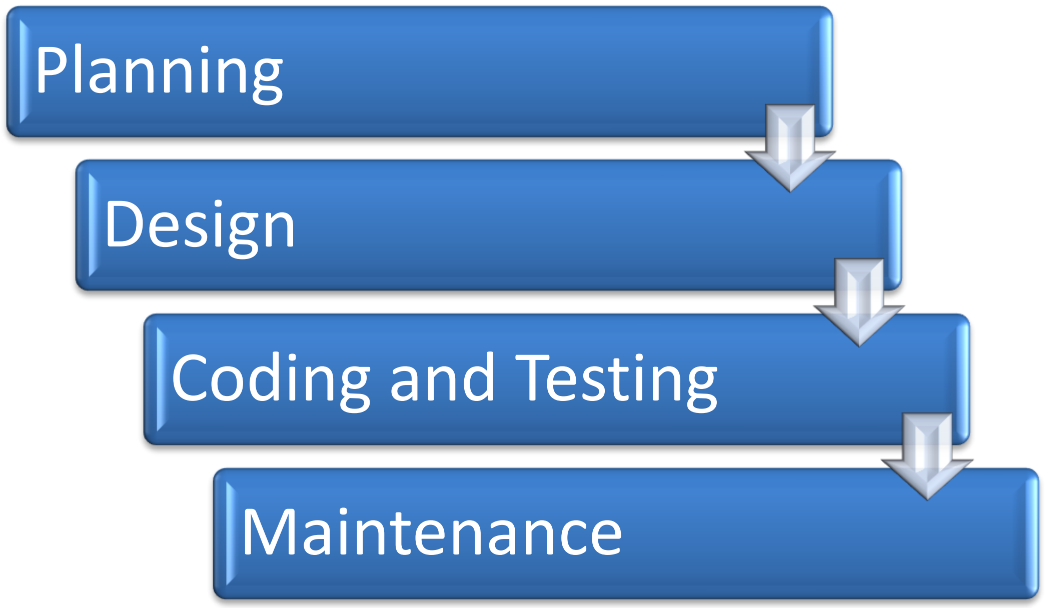


Figure 2: Development methodology

- Project planning

  In this phase, project planning has been done. Gantt chart, key milestones, project activities are defined here. All project planning has been planned within the timeline given. The project should be completed in FYP 2.

- Project design

  The analysis of the tool functionality is done in this phase. All research data that has been collected are gathered and analyzed for developing purposes. SWOT analysis is required in order to develop a tool that can be used for all testers.

- Coding and testing

  During this phase, development has been done in the beginning of FYP 2. The project is developed and tested simultaneously in order to save time usage. All tools required were used in this phase for development purposes.

- Evaluate tool

  Currently, maintenance for the tool developed is done. The tools will go through several testing phase in order to find any bugs or error. Finally, the tool will be reviewed by other user and should be completed in FYP 2.

The main code for this framework is coded in Java language with TestNG by using Eclipse IDE. In the main code contains all parameters and Selenese commands that will be used in test execution. It also includes the unit test part or the test script part which will define the web application URL and what features in the web application that wants to be tested.

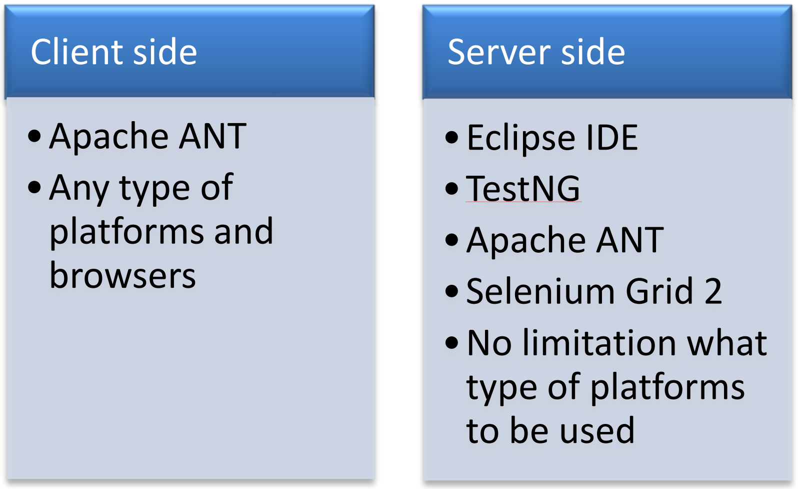
After done with the main code, the coding for test suite and parallel functionality is coded in XML language. Test suite is a collection of numbers of test cases that need to execute. This XML file is created for tester usage to choose what browsers that wants to run and wants to run in parallel or sequential. At this phase, the method to execute the test cases is by using Eclipse IDE.

When the testing by using Eclipse IDE is successful, the ANT build script is created. This file is a script for executing the test cases without using Eclipse IDE. The script is started with cleaning and removes the old reports in the report folder followed by building the test suite. The building test suite process is making the test suite directory. Lastly, this script will execute the test cases and locate the report of testing in the report folder.

In order to create framework for multiple testers, a JAR file or known as function library file is created. In this file, the main code and unit testing part is separated and the tester unable to edit or modify this main code. The testers only able to modify testgoogle.java file which contains unit testing part. The advantage of doing this is the testers no need to define all parameters from beginning. All the parameters already stored in JAR file.

3.3 Tools

Tools that will be used in this project:-



Client side	Server side
• Apache ANT • Any type of platforms and browsers	• Eclipse IDE • TestNG • Apache ANT • Selenium Grid 2 • No limitation what type of platforms to be used

Table 3: Tools will be used

Gantt chart has been planned throughout FYP 1 and FYP 2 proposed plan. Please refer **Figure 7** in **Appendix 1-1.**

CHAPTER 4

RESULTS AND DISCUSSIONS

4 RESULTS AND DISCUSSIONS

4.1 Results

This framework is developed to assist the software test engineers in automated web application testing especially for those who are not expert in programming because they can use and maintain this framework without difficulty and fast. This framework seems like a template with simple and basic programming language that can understand easily by drag and remove test cases created in a specific folder. From time to time, they can improve or enhance the framework due to increase the quality of automated testing.

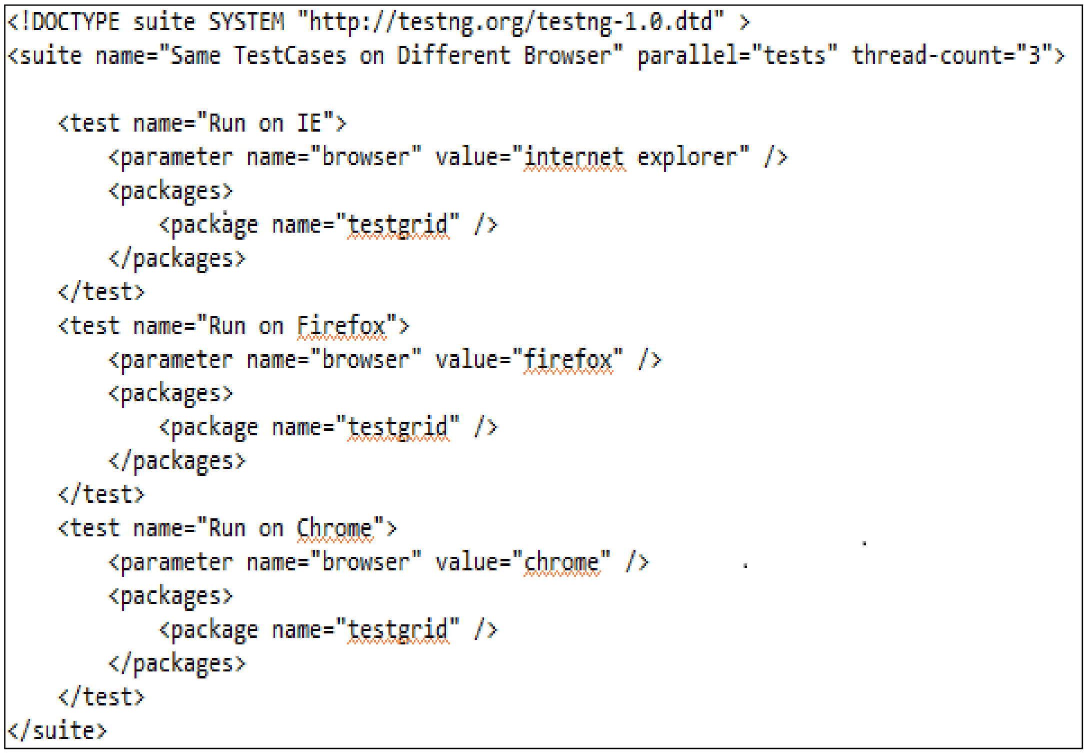
In this part will explain the details on how to use the framework that has been developed by using the improvement of Selenium. In order to save the execution time and automated tests with a group of testers, there are four files are created for this project.

i. *gridtestng.jar*

   This file is for function library and stored unit test function. It is used to launch webdriver based on browser manually defined in testng.xml file.

ii. *testng.xml*

It is a test suite (**Figure 3)** for test cases and it defines additional details in attributes. Tester can choose whether to run the test cases in parallel or not, and what types of browser or platform want to use.



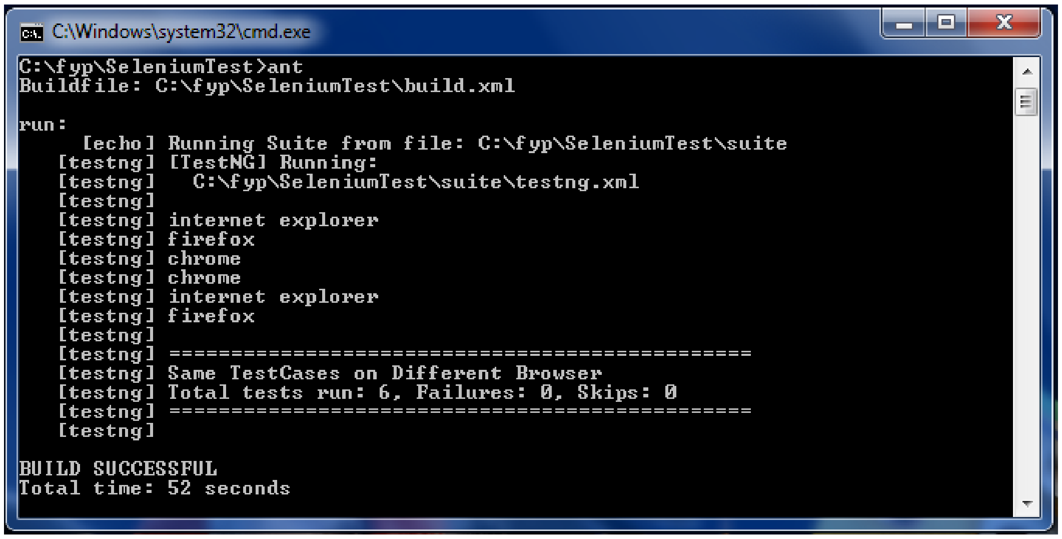
```xml
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Same TestCases on Different Browser" parallel="tests" thread-count="3">

    <test name="Run on IE">
        <parameter name="browser" value="internet explorer" />
        <packages>
            <package name="testgrid" />
        </packages>
    </test>
    <test name="Run on Firefox">
        <parameter name="browser" value="firefox" />
        <packages>
            <package name="testgrid" />
        </packages>
    </test>
    <test name="Run on Chrome">
        <parameter name="browser" value="chrome" />
        <packages>
            <package name="testgrid" />
        </packages>
    </test>
</suite>
```

Figure 3: Content of testng.xml file

iii. *build.xml*

This file contains build script (ANT script) for test execution and report generation against per browser type. This file will execute against the available test cases in the current folder. Tester need to execute the test cases by using command prompt in **Figure 4**. The report will display in HTML and XML format which is shown in **Figure 5** and **Figure 6**.

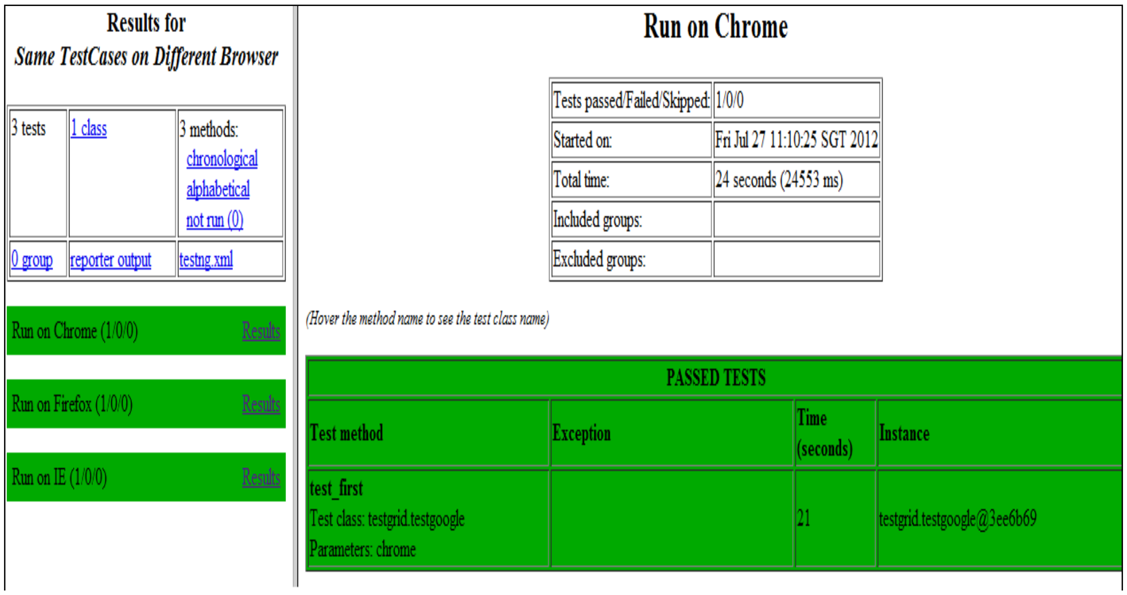


```
C:\fyp\SeleniumTest>ant
Buildfile: C:\fyp\SeleniumTest\build.xml

run:
     [echo] Running Suite from file: C:\fyp\SeleniumTest\suite
   [testng] [TestNG] Running:
   [testng]   C:\fyp\SeleniumTest\suite\testng.xml
   [testng]
   [testng] internet explorer
   [testng] firefox
   [testng] chrome
   [testng] chrome
   [testng] internet explorer
   [testng] firefox
   [testng]
   [testng] ===============================================
   [testng] Same TestCases on Different Browser
   [testng] Total tests run: 6, Failures: 0, Skips: 0
   [testng] ===============================================
   [testng]

BUILD SUCCESSFUL
Total time: 52 seconds
```

Figure 4: Executing test cases by using ANT



Results for
Same TestCases on Different Browser

3 tests	1 class	3 methods: chronological alphabetical not run (0)
0 group	reporter output	testng.xml

Run on Chrome (1/0/0) Results

Run on Firefox (1/0/0) Results

Run on IE (1/0/0) Results

Run on Chrome

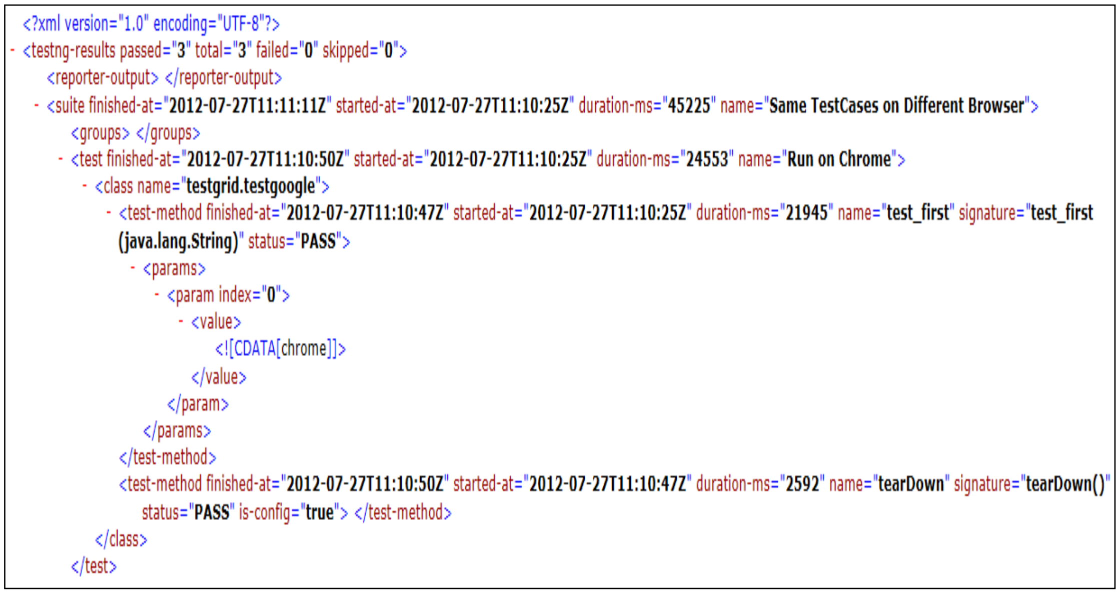
Tests passed/Failed/Skipped:	1/0/0
Started on:	Fri Jul 27 11:10:25 SGT 2012
Total time:	24 seconds (24553 ms)
Included groups:	
Excluded groups:	

(Hover the method name to see the test class name)

PASSED TESTS

Test method	Exception	Time (seconds)	Instance
test_first Test class: testgrid.testgoogle Parameters: chrome		21	testgrid.testgoogle@3ee6b69

Figure 5: Example for report display in HTML format

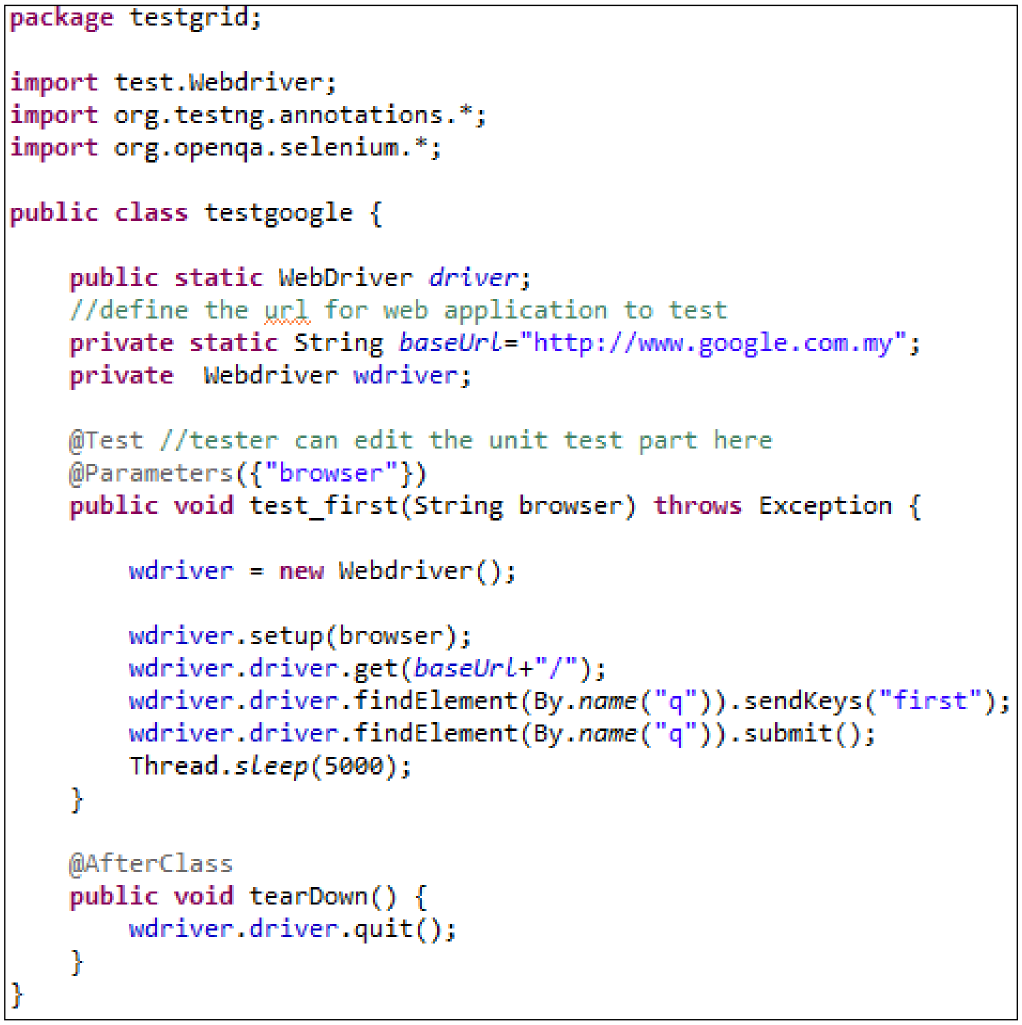


```
 <?xml version="1.0" encoding="UTF-8"?>
- <testng-results passed="3" total="3" failed="0" skipped="0">
    <reporter-output> </reporter-output>
  - <suite finished-at="2012-07-27T11:11:11Z" started-at="2012-07-27T11:10:25Z" duration-ms="45225" name="Same TestCases on Different Browser">
      <groups> </groups>
    - <test finished-at="2012-07-27T11:10:50Z" started-at="2012-07-27T11:10:25Z" duration-ms="24553" name="Run on Chrome">
      - <class name="testgrid.testgoogle">
        - <test-method finished-at="2012-07-27T11:10:47Z" started-at="2012-07-27T11:10:25Z" duration-ms="21945" name="test_first" signature="test_first
          (java.lang.String)" status="PASS">
          - <params>
            - <param index="0">
              - <value>
                  <![CDATA[chrome]]>
                </value>
              </param>
            </params>
          </test-method>
          <test-method finished-at="2012-07-27T11:10:50Z" started-at="2012-07-27T11:10:47Z" duration-ms="2592" name="tearDown" signature="tearDown()"
            status="PASS" is-config="true"> </test-method>
        </class>
      </test>
```

Figure 6: Sample report snippet in XML format

iv. *testgoogle.java*

This is a test case template and can be used by editing the unit test part. **Figure 7** is the example for test case used.



```
package testgrid;

import test.Webdriver;
import org.testng.annotations.*;
import org.openqa.selenium.*;

public class testgoogle {

    public static WebDriver driver;
    //define the url for web application to test
    private static String baseUrl="http://www.google.com.my";
    private  Webdriver wdriver;

    @Test //tester can edit the unit test part here
    @Parameters({"browser"})
    public void test_first(String browser) throws Exception {

        wdriver = new Webdriver();

        wdriver.setup(browser);
        wdriver.driver.get(baseUrl+"/");
        wdriver.driver.findElement(By.name("q")).sendKeys("first");
        wdriver.driver.findElement(By.name("q")).submit();
        Thread.sleep(5000);
    }

    @AfterClass
    public void tearDown() {
        wdriver.driver.quit();
    }
}
```

Figure 7: Sample test case that can be edited

This framework provides the additional following features from Selenium:

- Combines all Selenium Webdriver test cases and execute them using ANT script according to configurable parameters:-
    - Browsers type
    - Platforms type
- Executing all test cases in parallel per browser using ANT script. It can increase the test execution performance where no interface will be used.
- Provide TestNG HTML and XML report.

4.2 Discussions

This testing tool has been tested on different platforms. Each platform has different browsers to execute the testing. **Table 4** and **Table 5** show the result of testing that have done. For this testing, the number of test case use is two and it test the Google Search page.

	Windows	Linux	Mac
Internet Explorer	√	N/A	N/A
Firefox	√	√	√
Chrome	√	√	√

Table 4: Browsers and platforms compatibility

	Automated testing	Manual testing
Windows	52 seconds	124 seconds
Linux	45 seconds	85 seconds
Mac	48 seconds	119 seconds

Table 5: Time taken for different tests

Based on **Table 4**, all browsers and platforms are compatible with the testing tool. This testing tool is able to support to the latest versions of any browsers and platforms. In **Table 5** shows the difference of time taken according to automated and manual testing. For the automated testing, the times taken to execute the test cases are less than the manual testing which is less than one minute.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5 CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

It can be concluded that the first and second objectives in this project had been achieved from the developed framework. Thus, automated web application testing can reduce the time wastage doing manual testing. This framework can help software tester to maintain the tool more efficient and effective. The compatibility issue on executing test on Internet Explorer previously has been resolved. Currently, this testing tool is able to support all browsers version and platforms which are commonly used by Internet users.

5.2 Recommendations

For future work, this project is expected can distribute different test case to other machine and run in parallel at the same time so that it can increase the test execution process's performances. It can bring more benefits to the software testers who are implementing agile testing.

REFERENCES

Berner, S., Weber, R., & Keller, R. K. (2005). Observations and lessons learned from automated testing. *ICSE '05 Proceedings of the 27th international conference on Software engineering* (pp. 571-579). New York, USA: ACM.

Bolton, M. (2010). Rapid Software Testing. *Agile Conference*. Berlin.

Bucur, S., Ureche, V., Zamfir, C., & Candea, G. (2011). Parallel symbolic execution for automated real-world software testing. *EuroSys '11 Proceedings of the sixth conference on Computer systems* (pp. 183-198). New York, USA: ACM.

Cadar, C., Dunbar, D., & Englar, D. R. (2008). KLEE: Unassisted and automatic generation of high coverage test for complex system programs. *Symp. on Operating Systems Design and Implementation.* San Diego, CA: USENIX.

Candea, G., Bucur, S., & Zamfir, C. (2010). Automated software testing as a service. *SoCC '10 Proceedings of the 1st ACM symposium on Cloud computing* (pp. 155-160). New York, USA: ACM.

Ciupa, I., Meyer, B., Oriol, M., & Pretschner, A. (2008). Finding Faults: Manual Testing vs. Random+ Testing vs. User Reports. *19th International Symposium on Software Reliability Engineering* (p. 157). Zurich.

Gift, N. (2009, March 10). *Functional testing for Web applications*. Retrieved July 1, 2012, from IBM Developer Works: http://www.ibm.com/developerworks/web/library/wa-aj-testing/

Glover, A. (2006, August 29). *In pursuit of code quality: JUnit 4 vs. TestNG*. Retrieved April 2, 2012, from IBM Developer Works: http://www.ibm.com/developerworks/java/library/j-cq08296/

Horwath, T. (2007, January 18). *Testing Tool Evaluation Criteria*. Retrieved August 1, 2012, from Slideshare: http://www.slideshare.net/basma_iti_1984/testing-tool-evaluation-criteria-presentation

Kongsli, V. (2007). Security testing with Selenium. *OOPSLA '07 Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion* (pp. 862-863). New York, USA: ACM.

Li, Z., & Sun, Y. H. (2011, Jun 7). *Use Selenium Grid to enhance testing of web applications*. Retrieved April 2, 2012, from IBM Developer Works: http://www.ibm.com/developerworks/web/library/wa-seleniumgrid/index.html

Loreto, A. D. (2006, March 23). *Department of Computer Science.* Retrieved February 24, 2012, from The University of Western Ontario: http://www.csd.uwo.ca/courses/CS614b/DiLoreto1.pdf

Macnaghten, E. (2005, 12 16). *On free vs. proprietary.* Retrieved February 25, 2012, from Free Software Magazine: http://www.freesoftwaremagazine.com/node/1446

Mehaffy, D. W., Owen, W. W., & Aggarwal, A. (2010, August 31). *IBM's Test Automation Strategy: Build your test automation architecture around IBM Rational Quality Manager.* Retrieved Jun 9, 2012, from IBM Developer Works: http://www.ibm.com/developerworks/rational/library/10/build-test-automation-around-rational-quality-manager/

Ndem, G. C., Tahir, A., Ulrich, A., & Goetz, H. (2011). Test data to reduce the complexity of unit test automation. *OOPSLA '07 Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion* (pp. 105-106). New York, USA: ACM.

Open Source. (2012, February 25). *Test Automation*. Retrieved February 26, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Test_automation

Raman, V. N. (2006, March 21). *Sahi vs Selenium vs Watir.* Retrieved April 2, 2012, from Jungli Geek: http://narayanraman.blogspot.com/2006/03/sahi-vs-selenium-vs-watir.html

Razak, R. A., & Fahrurazi, F. R. (2011). Agile Testing with Selenium. *2011 5th Malaysian Conference in Software Engineering (MySEC),* (pp. 217-219). Kuala Lumpur.

Selenium Documentation Team. (2012). Selenium Documentation., 5.

Sirotkin, A. (2010). Web application testing with Selenium. *Linux Journal* , Article No. 3.

Sommerville, I. (2011). *Software Engineering.* United States of America: Pearson.

Sztipanovits, M., Qian, K., & Fu, X. (2008). The Automated Web Application Testing (AWAT) System. *ACM-SE 46 Proceedings of the 46th Annual Southeast Regional Conference on XX* (pp. 88-93). New York, USA: ACM.

Yu, P. C., & Chao, Z. C. (2010, December 7). *Automated web testing with Selenium.* Retrieved July 1, 2012, from IBM Developer Works: http://www.ibm.com/developerworks/opensource/library/os-webautoselenium/

APPENDICES

APPENDIX 1-1

Gantt Chart

		FYP 1				*FYP 2*				
NO	**PROCEDURE**	**JAN**	**FEB**	**MAR**	**APR**	**MAY**	**JUNE**	**JUL**	**AUG**	**SEPT**
1	Title selection/proposal	▨								
2	Submit proposal		▨							
3	Preliminary research work		▨	▨						
4	Preliminary Report (Extended Proposal)		▨							
5	Project work proceeds		▨	▨						
6	Viva			▨						
7	Project work proceeds			▨	▨					
8	Interim Report				▨					
9	Development phase				▨	▨				
10	Testing						▨	▨		
11	Implementation							▨	▨	
12	System Delivery								▨	▨

Figure 8: Gantt Chart

APPENDIX 2-1

Technical Paper

Automated Web Application Testing Using Improvement Selenium

Sharifah Anis Syed Naser
Information and Communication Technology Programme,
Universiti Teknologi PETRONAS
Bandar Seri Iskandar, Tronoh Perak, Malaysia
shanis.alkhadri@gmail.com

Any automated testing tool is useful if tester know how to maintain and use it properly. This paper will explain on how automated web application testing by using the improvement Selenium will initiate tester with improper technical skills in programming to maintain and reuse tools easily. The focus in this paper is to speed up the test execution process and improve the quality of Agile testing. Automated testing can be cost-effective testing tool if tester may have to be able to require the knowledge of the tool with different technologies.

Abstract- Agile Testing, regression testing, functional testing, technical skills, Selenium.

I. Introduction

Agile testing is a software testing practice for projects using agile methodologies [1]. Agile testing involves testing from the customer perspective as early as possible, testing early and often as code becomes available and stable enough, since working increments of the software are released often in agile software development. By using automated script to cut the cost and accelerate the whole test execution process is commonly done in agile testing. Agile testing is developed based on the theory that testers need to adapt to rapid development cycles and changes in testing patterns.

All software are developed and designed to meet and fulfil several functional requirements. A functional requirement can be technical, process or business based. Functional testing is the process by which expected behaviour of an application can be tested. Performing functional test for the application early in the software development cycle accelerate development, improves quality and reduces risks towards the end of the cycle. Mainly software goes through changes over a period of time. During the course of a single release, or across multiple releases, these changes may arise. Any changes to the software increase the chance of errors of bugs being introduce that breaks existing functionality is not broken is called regression testing. Test automation supports numerous regressions testing, frequently becoming a must in agile testing. Test automation is defined as using a software tool to run repeatable tests against the application to be tested [2]. Test automation provides responsiveness in regression testing.

In test automation, if there is a tester who is not expert in programming, he or she will have some difficulties in creating test cases where there is no test case template to use. If using the current testing tools, the group of testers will takes extra time because they have to create test cases from scratch. This problem is one of the agile testing challenges because it is involves all members of a cross-functional agile team. Some of testers have low initiative to improve their programming skills. They are preferred to use manually although it is time consuming. They did not realize that the power of programming skills in automated testing can change the quality of agile testing. The purpose of this paper is to explain how this framework can handle automated tests with a group of testers. This framework can ease the testers and it is no limitation on what type of browsers and platforms to use. They can configure the parameters according to the testing requirements.

This paper is divided into four sections, including introduction which is section 1. In section 2 is about background of study. Section 3 includes methodology and section 4 includes results and discussions. The last section 5 will be the conclusion and future work.

II. Background of Study

In this section will explain briefly about Selenium. The relevant literature review that related with this project also included for the background information on the research and to identify what others have said and discovered about the research.

A. Selenium

In order to support test automation, Selenium is the best tool that can be used which is a suite of tools to automate web browsers across many platforms. Selenium is an open source tool can runs in many type of browsers and platforms that are commonly used by internet users. It can be controlled by many testing frameworks and programming languages (e.g Java, Ruby, Python, C#, Perl, PHP and Groovy). Besides that, it composed of multiple software tools and each tool has a specific role.

Selenium 2 (aka. Selenium Webdriver) is the enhancement of Selenium 1 which accepts commands and sends them to a browser. This is done through a browser-specific browser driver, which sends commands to a browser, and retrieves results. Selenium 2 executed tests without a special server like Selenium 1 but it directly starts a browser instance and controls it.

Selenium 1(aka. Selenium RC or Remote Control) is a server that accepts commands for the browser via HTTP. When launching html test case, a new instance of Selenium server is needed. It means that the port cannot be same for each parallel run.

Selenium IDE is a Firefox extension that allows recording, editing and debugging tests in a short time. Previously it was known as Selenium Recorder.

Selenium Grid is a server that allows tests to use web browser instances running on remote machines. One server acts as the hub and it has a list of servers that provide access to browser instance (nodes), and lets tests use these instances. Selenium Grid supports to run tests in parallel on multiple machines, and to control different browser versions [3].

B. Literature Review

Programming skills are vital in automation where the tester need to understand the problem or errors found and debug it in the system when they find the problems. Maintainable and reusable tool building requires good programming skills and knowledge of the tool (Loreto, 2006) [4]. Loreto try to explain that automation requires technical skills where in software testing nowadays, still have testers who are not expert in programming skills. This problem can be a great challenge to an organization implementing agile methodology in their software projects. While, Sztipanovits, et al.,(2008) agreed with Loreto's statement that currently the time and labor cost of grading of tester web application assignments is a shared issue for all related programming courses [5]. Horwarth (2007) come out with his support statement saying that most software testing projects do fail because one of the reasons is test technicians with improper skills are assigned to use these automated test tools. Users of these tools must have strong test mentalities and in all but a few situations they must also possess solid programming skills with the automation tool's scripting language [6].

In this era, all systems are developed as an automated due to save more time and cut the cost of production. Thus, having a great programming skills are required in order to fully utilize automated testing. For this project, the type of programming skills that will be used is Java which is easy to use and easy to find reference if any problems happened. Also, a template of test case which contains the basic line of codes that is useful for tester to use in their project testing. Tester can edit or modify the template according to what they want. From here, tester can learn and improve their programming skills slowly by modifying test case.

III. Methodology

From the interview sessions that have been done, some of the testers are still using manual testing in certain testing. The reason why they are still using manual testing is not all of them are very expert in programming which is required in creating automated scripts. Most of them are familiar and know about Selenium but not all of them have a deep knowledge. Some of them either only know the basic techniques of using Selenium or do not know how to use Selenium. Moreover, they realize that using manual testing is consuming more time and cost.

Figure 1 is the framework designed for this project by using Selenium Grid 2. The format of test case is Selenium Webdriver in Java language. There is no limitation for the number of test cases that will be used during test execution.

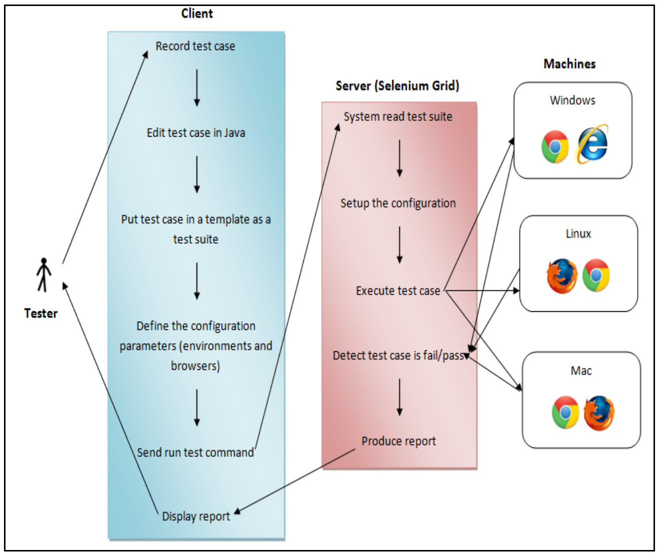


Figure 1: Framework design

IV. Results and Discussions

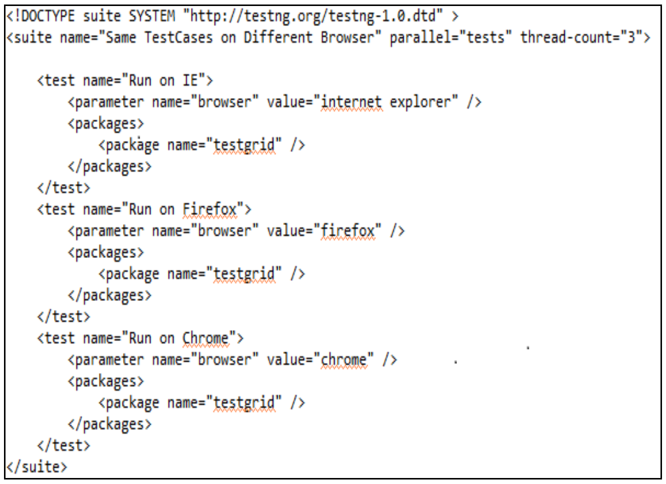
In this part will explain the details on how to use the framework that has been developed by using the improvement of Selenium. There are four files are created for this project:-

a) gridtestng.jar

This file is for function library and stored unit test function. It is used to launch webdriver based on browser manually defined in testng.xml file.

b) testng.xml

It is a test suite for test cases and it defines additional details in attributes. Tester can choose whether to run the test cases in parallel or not, and what types of browser or platform want to use. Figure 2 shows the content of testng.xml file.



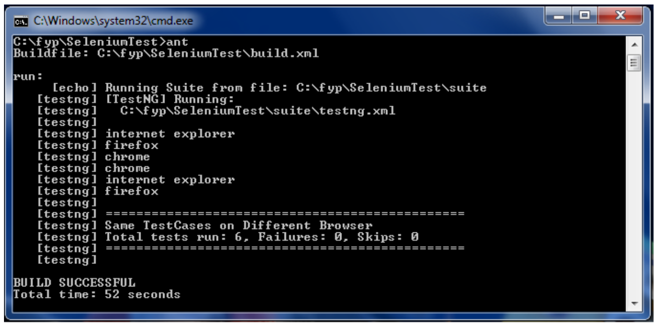
```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Same TestCases on Different Browser" parallel="tests" thread-count="3">

    <test name="Run on IE">
        <parameter name="browser" value="internet explorer" />
        <packages>
            <package name="testgrid" />
        </packages>
    </test>
    <test name="Run on Firefox">
        <parameter name="browser" value="firefox" />
        <packages>
            <package name="testgrid" />
        </packages>
    </test>
    <test name="Run on Chrome">
        <parameter name="browser" value="chrome" />
        <packages>
            <package name="testgrid" />
        </packages>
    </test>
</suite>
```

Figure 2: Content of testng.xml file

c) build.xml

This file contains build script (ANT script) for test execution and report generation against per browser type. This file will execute against the available test cases in the current folder. Tester need to execute the test cases by using command prompt as shown in Figure 3. The report will display in HTML (Figure 4) and XML (Figure 5) format.

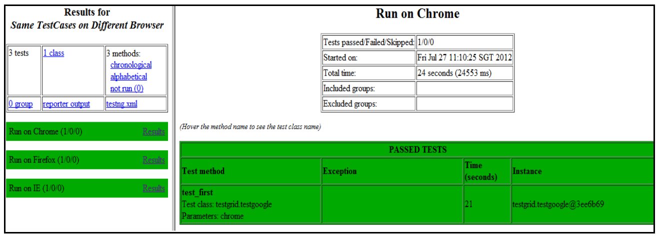


```
C:\fyp\SeleniumTest>ant
Buildfile: C:\fyp\SeleniumTest\build.xml

run:
     [echo] Running Suite from file: C:\fyp\SeleniumTest\suite
   [testng] [TestNG] Running:
   [testng]   C:\fyp\SeleniumTest\suite\testng.xml
   [testng]
   [testng] internet explorer
   [testng] firefox
   [testng] chrome
   [testng] chrome
   [testng] internet explorer
   [testng] firefox
   [testng]
   [testng] ===============================================
   [testng] Same TestCases on Different Browser
   [testng] Total tests run: 6, Failures: 0, Skips: 0
   [testng] ===============================================
   [testng]

BUILD SUCCESSFUL
Total time: 52 seconds
```

Figure 3: Executing test cases by using ANT



Results for
Same TestCases on Different Browser

3 tests	1 class	3 methods: chronological alphabetical not run (0)
0 group	reporter output	testng.xml

Run on Chrome (1/0/0) Results

Run on Firefox (1/0/0) Results

Run on IE (1/0/0) Results

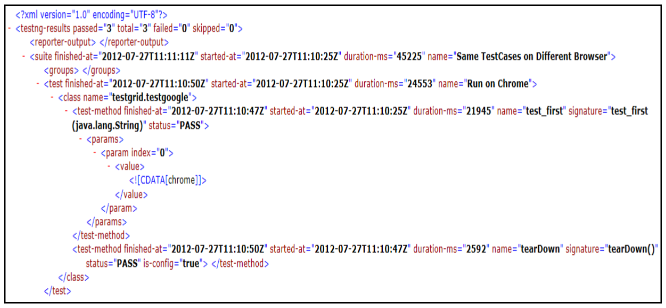
Run on Chrome

Tests passed/Failed/Skipped:	1/0/0
Started on:	Fri Jul 27 11:10:25 SGT 2012
Total time:	24 seconds (24553 ms)
Included groups:	
Excluded groups:	

(Hover the method name to see the test class name)

PASSED TESTS			
Test method	**Exception**	**Time (seconds)**	**Instance**
test_first Test class: testgrid.testgoogle Parameters: chrome		21	testgrid.testgoogle@3ee6b69

Figure 4: Example report display in HTML format

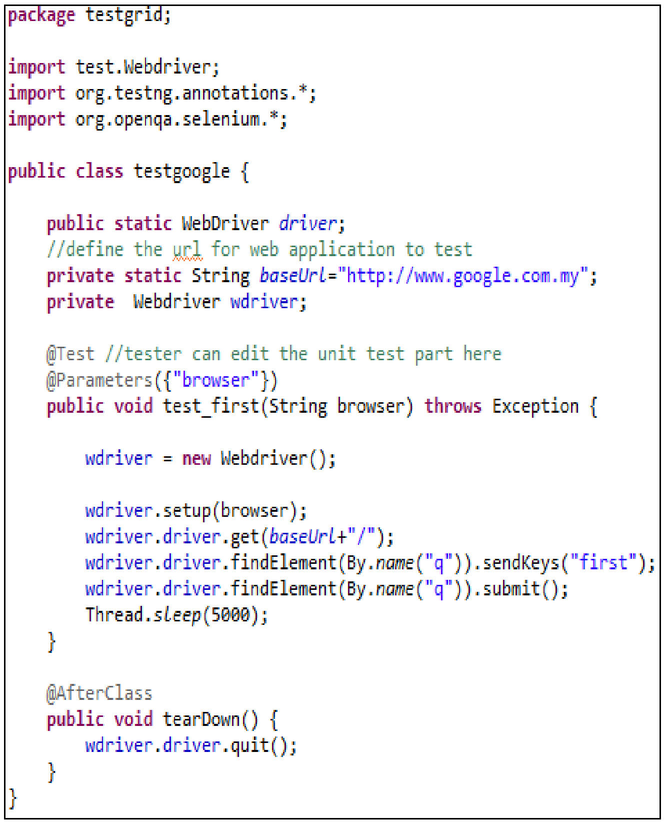


```
  <?xml version="1.0" encoding="UTF-8"?>
- <testng-results passed="3" total="3" failed="0" skipped="0">
    <reporter-output> </reporter-output>
  - <suite finished-at="2012-07-27T11:11:11Z" started-at="2012-07-27T11:10:25Z" duration-ms="45225" name="Same TestCases on Different Browser">
      <groups> </groups>
    - <test finished-at="2012-07-27T11:10:50Z" started-at="2012-07-27T11:10:25Z" duration-ms="24553" name="Run on Chrome">
      - <class name="testgrid.testgoogle">
        - <test-method finished-at="2012-07-27T11:10:47Z" started-at="2012-07-27T11:10:25Z" duration-ms="21945" name="test_first" signature="test_first
          (java.lang.String)" status="PASS">
          - <params>
            - <param index="0">
              - <value>
                  <![CDATA[chrome]]>
                </value>
              </param>
            </params>
          </test-method>
          <test-method finished-at="2012-07-27T11:10:50Z" started-at="2012-07-27T11:10:47Z" duration-ms="2592" name="tearDown" signature="tearDown()"
            status="PASS" is-config="true"> </test-method>
        </class>
      </test>
```

Figure 5: Example report snippet display in XML format

d) testgoogle.java

This is a test case template and can be used by editing the unit test part. Figure 6 is the example for test case used.



```
package testgrid;

import test.Webdriver;
import org.testng.annotations.*;
import org.openqa.selenium.*;

public class testgoogle {

    public static WebDriver driver;
    //define the url for web application to test
    private static String baseUrl="http://www.google.com.my";
    private  Webdriver wdriver;

    @Test //tester can edit the unit test part here
    @Parameters({"browser"})
    public void test_first(String browser) throws Exception {

        wdriver = new Webdriver();

        wdriver.setup(browser);
        wdriver.driver.get(baseUrl+"/");
        wdriver.driver.findElement(By.name("q")).sendKeys("first");
        wdriver.driver.findElement(By.name("q")).submit();
        Thread.sleep(5000);
    }

    @AfterClass
    public void tearDown() {
        wdriver.driver.quit();
    }
}
```

Figure 6: Sample test case

This framework provides the additional following features from Selenium:-

a) Combines al Selenium Webdriver test cases and execute them using ANT script according to configurable parameters for browsers type and platforms type.
b) Executing all test cases in parallel per browser using ANT script. It can increase the test execution performance where no interface will be used.
c) Provide TestNG HTML and XML reports.

V. Conclusion and Future work

Automated web application testing can reduce the time wastage when a tester needs to perform a certain amount of testing in a limited time. This framework can help software tester to maintain the tool more efficient and effective. At the same time, it is able to assist the testers to improve their programming skills for the tool maintenance.

For the future work this framework can be extended with the functionality of capturing screenshot when the test failed. This functionality can improve more the quality of testing.

VI. References

[1] Bolton, M. (2010). Rapid Software Testing. Agile Conference. Berlin.

[2] Open Source. (2012, February 25). "Test Automation" Retrieved February 26, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Test_automation

[3] Selenium Documentation Team. (2012). "Selenium Documentation". - , 5.

[4] Loreto, A. D. (2006, March 23). Department of Computer Science. Retrieved February 24, 2012, from The University of Western Ontario: http://www.csd.uwo.ca/courses/CS614b/DiLoreto1.pdf

[5] Sztipanovits, M., Qian, K., & Fu, X. (2008)." The Automated Web Application Testing (AWAT) System". ACM-SE 46 Proceedings of the 46th Annual Southeast Regional Conference on XX (pp. 88-93). New York, USA: ACM.

[6] Horwath, T. (2007, January 18). "Testing Tool Evaluation Criteria". Retrieved August 1, 2012, from Slideshare: http://www.slideshare.net/basma_iti_1984/testing-tool-evaluation-criteria-presentation